实用第一 智慧密集 Practicability First, Intelligence Intensive

电脑编程技巧与维护

COMPUTER PROGRAMMING SKILLS & MAINTENANCE

http://www.comprg.com.cn

上 4月 2013年4月03日

每期定价:11.00元 全年定价:264.00元
《电脑编程技巧与维护》杂志社出版
刊号: ISSN 1006-4052 CN11-3411/TP
广告许可证 京海工商广字0151

国家级科技期刊 中国学术期刊综合评价数据库统计源期刊 中国核心期刊（遴选）数据库收录期刊

ISSN 1006-4052

汇主流编程语言　精选典型编程案例

聚宝贵编程经验　揭示编程技术诀窍

《电脑编程技巧与维护》杂志社　编著

定价：79元

订阅方式：

汇款地址：北京市海淀区长春桥路5号6号楼1209室　　收款人：电脑编程技巧与维护杂志社　　邮编：100089

杂志社官方淘宝店：http://compre.taobao.com　　电话/传真：82561614

E-mail:zzs**@vip.sina.com　　QQ:565699495

汇款如未注明所购买数量和邮寄地址，请与杂志社联系。

2013 年第 07 期
4 月(上)

电脑编程技巧与维护

总第 277 期　1994 年 7 月创刊　(半月刊)

地　　址：北京市海淀区长春桥路 5 号
6 号楼 1209 室
投 稿 信 箱：gaojian@comprg.com.cn
gaojian@comprg.sina.net
编辑部信箱：gaojian@comprg.com.cn
发行部信箱：zzsfx@vip.sina.com
网　　址：http://www.comprg.com.cn
邮　　编：100089
电　　话：(010) 82561037
传　　真：(010) 82561614
照　　排：《电脑编程技巧与维护》
杂志社电脑排版部
印 刷 厂：北京慧美印刷有限公司
订　阅　处：全国各地发行局
国内总发行：北京报刊发行局
邮 发 代 号：82-715
国外发行代号：M6232
刊　　号：ISSN 1006-4052 / CN11-3411/TP
广告订可证：京海工商广字 0151 号
全 年 定 价：264 元
每 期 定 价：11 元

电脑编程技巧与维护(半月刊)

每月3日出版

实用第一 智慧密集

目次

可让手机成为 3D 扫描仪的智能手机软件即将问世

据国外媒体报道一款即将问世叫 “Moedls” 的手机应用软件，通过 iPhone 手机或 iPad 电脑，以及一个商用激光器、转盘和简单的盒子，便能实现物体的 3D 扫描。发明者是约翰-费尔，他曾成功设计磁性弹弓和升空磁性雕像。

费尔说：“3D 扫描难度较大的部分实际是转盘，很难发现一个平台旋转缓慢，并且足够稳定。在此之前我曾使用过一个老式留声机。”

将一部智能手机放置在三脚架上，把需要扫描的物体放在转盘上，一个小型激光器瞄准扫描物体，之后激光器照亮进行扫描。激光器的功能有点儿像照相机闪光灯，提供一个真实明亮的光源，使相机能够拍摄到扫描物体的细节部分。当扫描物体在转盘上转动，智能手机相机能够拍摄更多照片，最终获得 3D 图像结构。目前，这款手机软件仍在等待苹果公司的审批，如果获批将出现在 iPhone 手机的应用商店中，同时，该软件的手机安卓版正在研发之中。

触宝输入法新功能：滑动输入状态下支持整句长句识别

在刚结束的巴塞罗那 MWC 大会上，触宝 TouchPal 作为国内少有参加 MWC 展会的软件开发者亮相了他们即将正式推出的新版 TouchPal 输入法国际版，新版本的扫屏滑动输入将更“闪电”，通过西文的字频、词频及用户习惯，用户能够在手指不离开屏幕的情况下输入完整长句。

在 2011 年推出的 TouchPal 是 “增强版的 Swype” + SwiftKey（SwiftKey 后来也推出了滑动输入 SwiftKey Flow），TouchPal 的专利 “TouchPal Curve?” 可让用户在不抬起手指的情况下输入单词，支持盲打纠错。同时它有强大的联想输入功能，通过上下文和用户行为预测词汇。

而新增的 TouchPal Wave 功能将联想功能发挥得更好。和 BB 10 发布会上虚拟键盘的词汇预测输入功能 “writing without typing” 有点像，在使用 TouchPal 输入法新版本做输入的时候，句子的下一个预测词会将会显示在可能会滑到的字母上方，如输入 “looking”，f、u、l、a 四个字母上方会分别显示 “forward”、“up”、“like”、“at”。如果用户下一个想打的词是 forward，用户将手指移至 f 键并向空格键滑行，forward 即会被输入，后续联想 “to seeing you” 只要手指移至有相关联想词的字母处便可打出，手指不离开屏幕就能输入长句。这个更强大的滑动输入功能是 TouchPal 第一个做出来的。

根据触宝的数据，触宝输入法已经成为中兴、华为、HTC、索尼等制造商的默认键盘，支持 70 种语言，在国际上有一定份额。Android 版本的 TouchPal 输入法就占了 Android 20%的市场，而他们最大的竞争对手的 Swype 约占 70%的市场（曾经被 Nuance 以 1 亿美元收购）。据触宝透露，这次的 MWC 大会让他们有了更多向制造商展示的机会，他们未来可能将会与更多知名大型厂商建立合作。之前触宝团队曾表示会研究更适合中国国情的中文滑行输入，但不会是简单的测试直接把西文滑行方式借鉴过来。

人类 10 到 15 年内拥有全息平台

每个科幻爱好者和痴迷技术的人儿都希望能拥有一个全息成像台。不幸的是，制造全息平台的技术还尚未被人类掌握，据不太可靠信息说这项技术可用之时还有大约 10 到 15 年的时间——这是 AMD 的专业人士 Phil Rogers 说的，他专攻 3D 技术工作已超过 20 年。

在一次访谈节目的对话中，Rogers 列出了在全息投影进入应用之前的 7 个必须被克服的障碍：

1.比 IMAX 更好的视觉体验

全息成像台需要 360 度保真。并且它还得明白，当观影者走近时画面上的物体也要变近，同理当观影者走远时画面也变远。当观影者转头时画面会产生相应的倾斜；当同时显示多个视频源时，每幅画面都要完美地互相拼接。

2.绝无仅有的高保真的音效

为了让全息的用户体验更加真实，那么音频不仅要是沉浸式的，并且在不同的方向或声场中用户听到的效果也有所区别。

3.触感

“我们还需要继续发展触觉反馈，比如一个全息影像中的人是尝试与一个物品互动，他们需要感觉到自己是真的触碰到了它们，” Rogers 说。目前最可行的方法就是定向空气喷射。

4.高效的内存分配

“要保证全息平台运作正常稳定，最好的方法就是实现 CPU 和 GPU 内存共享，” Rogers 说。在这个方向上我们目前的进步还是挺快的，不过要达到同时实现百万级的并行处理（parallel precesses），还有很长的路要走。

5.超强的处理能力

要动用百万台级别的电脑才能同步所有的图像，视频以及其他保证全息平台真实性的内容，“问题就是我们需要百万台甚至更多主机级别的电脑，” Rogers 指出，“这就是说要等到全息平台在商业上有可运作性时，所采用的原件不会有如此大的数量级和能耗。”

6.找到付费客户

当然，如果它没有购买价值的话也很难在市场上出现。企业认为商务视频会议会从这项技术中受益很多，并且到时候可能不再需要费时费力的 PPT 制作。

7.目标瞄准开源社区

网上默默无闻却潜力无穷的技术人员是很多的。AMD 计划将全息平台的建筑方案留给了网上的业余程序员们。

iBatis.Net（C#）系列二：SQL 数据映射

张德强　祁亚玲

摘　要：探讨了 iBatis.Net 框架的 XML 数据映射文件各配置节点的含义，并通过对数据库的 CRUD 4 种操作讲解了配置数据映射文件和调用方法。

关键词：iBatis.Net 框架；XML 数据；SQL Maps 架构；数据映射

上篇介绍了 iBatis.Net 的基本情况和运行原理，运行环境中各参数的配置情况。并通过一个实例项目进行了说明。

1　数据映射基础

SQL Maps 是这个 iBatis.Net 框架中最重要的部分，而 SQL Maps 的核心就在于 XML 数据映射文件 (Data Map XML File)。在 XML 数据映射文件里可以定义包括要执行各种 SQL 语句、存储过程、输入参数映射、返回结果映射、缓存机制，并且能通过几种相对比较复杂的配置实现对象之间的关联关系和延迟加载。这也是 iBatis.Net 区别其他 ORM 框架而具备更灵活性，更高性能的关键所在。

配置文件可以写得很简单，也可以很复杂。复杂配置文件往往出于更好的设计，更好性能，更好扩展性方面的目的。一个 XML 数据映射文件主要分为两个部分：模块配置和语句配置。

2　模块配置

2.1　Type Alias 节点

定义一个 XML 数据映射文件中的别名，其好处就是避免过长变量值的反复书写，比如通过 typeAlias 节点为类“iBatisTest.Domain.Sysuser”定义了一个别名“Sysuser”,这样在这个数据映射文件中的其他部分，需要引用“iBatisTest.Domain.Sysuser”类时，只需以其别名替代即可，和 SqlMap.config 配置文件里面的 Alias 节点配置一样。定义如：

```
<alias ><typeAlias  alias = "Sysuser"  type = "iBatisTest.Domain.Sysuser,iBatisTest" /></alias>
```

2.2　cacheModel 节点

缓存机制是程序开发中经常讨论的一个话题，在实际的系统开发过程中，总会存在着这样一类数据，它们更新频率很低，然而使用的频率却非常之高。为了提高系统性能，通常将此类数据装入缓存。iBatis.Net 也有自己的缓存系机制，它通过 cacheModel 节点来配置，具体的定义如下：

```
<cacheModel type="LRU" id="Sysuser-cache"  readOnly="true" serialize="false">
<flushInterval hours="24"/>
<flushOnExecute statement="SysuserMap. UpdateSysuser "/>
>
<property name="CacheSize" value="100"/>
</cacheModel>
```

type：缓存的类型，iBatis.Net 中有 4 种类型，分别为 MEMORY、LRU、FIFO、OSCACHE。

其中 MEMORY 是内存缓存，LRU 是使用最近最少使用策略，FIFO 是使用先进先出策略，OSCACHE 是通过第三方的缓存插件来实现。

id：是 cacheModel 的一个标识，标识该缓存的名字，供后面设置使用。

readOnly：指缓存的数据对象是只读还是可读写，默认为“true”，即只读，这里的只读并不是意味着数据对象一旦放入缓存中就无法再对数据进行修改。而是当数据对象发生变化的时候，如数据对象的某个属性发生了变化，则此数据对象就将被从缓存中废除，下次需要需重新从数据库读取数据，构造新的数据对象。而 readOnly=" false" 则意味着缓存中的数据对象可更新。

serialize：是否从缓存中读取同一个对象，还是对象的副本。该参数只有在 readOnly 为 false 的情况下才有效，因为缓存是只读的，那么为不同会话返回的对象肯定是一个，只有在缓存是可读写的时候，才需要为每个会话返回对象的副本。

flushInterval：指定缓存自动刷新的时间，可以为 hours、minutes、seconds 和 milliseconds。需要注意的是，这个间隔时间不是时间到了，在缓存里的信息会自动刷新，而是在间隔时间过后，下次查询将不会从缓存中去取值，而会用 SQL 去查询，同时将查询的结果更新缓存的值。

flushOnExecute：指定在发生哪些操作时，更新缓存。

property：针对 cacheModel 的额外的一些属性配置，不同 type 的 cacheModel 将会有自己专有的一些 property 配置，如：

FIFO: <property name="CacheSize" value="100" />
LRU: <property name="CacheSize" value="100" />
MEMORY: <property name="Type" value="WEAK" />

当配置好 cacheModel 之后就可以在后面的语句中使用了，

如：

```
< statement   id="SelectSysuser" resultClass="Sysuser"
cacheModel="Sysuser-cache" >
        SELECT  *  FROM DEAN.SYSUSER
< /statement >
```

2.3 resultMap 节点

resultMap 从字面理解就是结果集的映射，它将返回的记录与实体对象进行映射，它属于直接映射。如果查询出来的数据集的字段与属性和实体类一致时，常用 resultClass 来替代，它属于隐身映射。通常 resultMap 比 resultClass 性能要高。

在使用 resultMap 时需要注意，如果在 resultMap 中给出的配置字段，但是返回的数据集却没有返回这个字段，那程序将给出异常。相反的，如果返回了一些字段，却没有在 resultMap 给出配置定义的话，那么那些字段将不会被处理而不会给出任何的提示，相当没有查询出这些字段。具体的定义例子如下：

```
<resultMaps>
  <resultMap id="SysuserResult" class="Sysuser">
<result property="Userid" column="USERID" />
<result property="Password" column="PASSWORD" />
<result property="Loginname" column="LOGINNAME" />
<result property="Sex" column="SEX"/>
<result property="Birthday" column="BIRTHDAY"/>
<result property="Idcard" column="IDCARD" />
<result property="Officephone" column="OFFICEPHONE" />
<result property="Familyphone" column="FAMILYPHONE" />
<result property="Mobilephone" column="MOBILEPHONE"/>
<result property="Email" column="EMAIL"/>
<result property="Address" column="ADDRESS"/>
<result property="Zipcode" column="ZIPCODE"/>
<result property="Remark" column="REMARK" />
<result property="Status" column="STATUS" />
<result property="Registerdate" column="REGISTERDATE" />
  </resultMap>
</resultMaps>
```

该参数也是查询语句特有的配置项，因为 insert、update、delete 3 种语句的操作是不会返回数据结果集的，也就没有 resultMap 和 resultClass。如果在一个语句配置中同时指定了 resultMap 和 resultClass 属性的话，将会优先使用 resultMap。同时 result Map 的使用也是一个实现对象复杂查询功能的重要手段，如：result map 的继承，对象的 1..1、1..N 关系查询。

2.4 ParameterMap 节点

参数映射的配置，它是被用来向一个语句提供所需参数的配置。该参数配置节点和 resultMaps 节点类似。

3 语句配置

语句配置（Mapped Statements）：顾名思义就是映射的语句声明。它是 XML 数据映射文件的核心，在 iBatis.Net 框架中真正和数据库打交道的被执行的 SQL 语句（或存储过程）都必须在这里被显式声明。语句配置可以包含有：statement、select、insert、update、delete 和 procedure 这 6 种不同的语句类型。其中 statement 可以包含所有类型的 SQL 语句（存储过程），它是一个泛泛的语句配置，没特别明确的职责，相反，其他 5 种类型的语句配置就是专门负责各种不同的 SQL 语句。表 1 列出了 6 种类型的语句的不同职责和调用方法。

表 1 语句配置中的不同语句类型

Statement 类型	属性	子元素	方法
<statement>	id parameterClass resultClass parameterMap resultMap cacheModel	所有动态元素	insert update delete 所有的查询方法
<insert>	id parameterClass parameterMap	所有的动态元素 <selectKey>	insert update delete
<update>	id parameterClass parameterMap	所有的动态元素	insert update delete
<delete>	id parameterClass parameterMap	所有的动态元素	insert update delete
<select>	id parameterClass resultClass parameterMap resultMap cacheModel	所有的动态元素	所有的查询方法
<procedure>	id parameterClass resultClass parameterMap resultMap	所有的动态元素	insert update delete 所有的查询方法

应用系统操作数据库数据一般有 CRUD 4 种方式，即增加 (Create)、查询 (Retrieve，重新得到数据)、更新 (Update) 和删除 (Delete)。下面详细介绍一下这 4 种情况的 SQL 语句配置及调用方法。

首先在 Oracle 数据库中新建表 DEAN.SYSUSER。表结构如图 1 所示。

	Name	Code	Data Type	Length
1	登录ID	USERID	number(10,0)	10
2	登录密码	PASSWORD	VARCHAR2(40)	40
3	登录用户名	LOGINNAME	VARCHAR2(40)	40
4	性别	SEX	VARCHAR2(40)	40
5	出生日期	BIRTHDAY	DATE	
6	身份证号	IDCARD	VARCHAR2(18)	18
7	办公电话	OFFICEPHONE	VARCHAR2(40)	40
8	家庭电话	FAMILYPHONE	VARCHAR2(40)	40
9	手机	MOBILEPHONE	VARCHAR2(40)	40
10	Email	EMAIL	VARCHAR2(100)	100
11	通讯地址	ADDRESS	VARCHAR2(200)	200
12	邮编	ZIPCODE	VARCHAR2(20)	20
13	备注	REMARK	VARCHAR2(200)	200
14	状态	STATUS	VARCHAR2(20)	20
15	注册时间	REGISTERDATE	DATE	

图 1 SYSUSER 表结构

增加实体类 iBatisTest.Domain.Sysuser，该实体类和表DEAN.SYSUSER 对应。Sysuser 类代码如下：

```
using System;
namespace iBatisTest.Domain
```

FOLLOW MASTER PROGRAM

```
{
 [Serializable]
 public sealed class Sysuser
 {
        #region 私有成员定义
        private Int32 _userid;
        private string _password;
        private string _loginname;
        private string _sex;
        private DateTime _birthday;
        private string _idcard;
        private string _officephone;
        private string _familyphone;
        private string _mobilephone;
        private string _email;
        private string _address;
        private string _zipcode;
        private string _remark;
        private string _status;
        private DateTime _registerdate;
        #endregion 私有成员定义
        #region 定义默认类结构函数
        public Sysuser()
        {
            _userid = 0;
            _password = null;
            _loginname = null;
            _sex = null;
            _birthday = new DateTime();
            _idcard = null;
            _officephone = null;
            _familyphone = null;
            _mobilephone = null;
            _email = null;
            _address = null;
            _zipcode = null;
            _remark = null;
            _status = null;
            _registerdate = new DateTime();
        }
        #endregion 定义默认类结构函数
        #region 定义公有属性
        /// <summary>
        /// 登录 ID
        /// </summary>
        public Int32 Userid
        {
            get { return _userid; }
            set { _userid = value; }
        }
        /// <summary>
        /// 登录密码
        /// </summary>
        public string Password
        {
            get { return _password; }
            set
            {
                if( value! = null && value.Length > 40)
                throw new ArgumentOutOfRangeException("密码
值错误", value, value.ToString());
                _password = value;
            }
        }
        /// <summary>
        /// 登录用户名
        /// </summary>
        public string Loginname
        {
            get { return _loginname; }
            set
            {
                if( value! = null && value.Length > 40)
                throw new ArgumentOutOfRangeException("登
录用户名值错误", value, value.ToString());
                _loginname = value;
            }
        }
        /// <summary>
        /// 性别
        /// </summary>
        public string Sex
        {
            get { return _sex; }
            set
            {
                if( value! = null && value.Length > 40)
                throw new ArgumentOutOfRangeException("性别值
错误", value, value.ToString());
                _sex = value;
            }
        }
        /// <summary>
        /// 出生日期
        /// </summary>
        public DateTime Birthday
        {
            get { return _birthday; }
            set { _birthday = value; }
        }
        /// <summary>
        /// 身份证号
        /// </summary>
        public string Idcard
```

```
        {
            get { return _idcard; }
            set
            {
                if( value!  = null && value.Length > 18)
                throw  new  ArgumentOutOfRangeException ("
身份证号值错误", value, value.ToString());
                _idcard = value;
            }
        }
        /// <summary>
        /// 办公电话
        /// </summary>
        public string Officephone
        {
            get { return _officephone; }
            set
            {
                if( value!  = null && value.Length > 40)
                throw new ArgumentOutOfRangeException("
办公电话值错误", value, value.ToString());
                _officephone = value;
            }
        }
        /// <summary>
        /// 家庭电话
        /// </summary>
        public string Familyphone
        {
            get { return _familyphone; }
            set
            {
                if( value!  = null && value.Length > 40)
               throw new ArgumentOutOfRangeException("家
庭电话值错误", value, value.ToString());
                _familyphone = value;
            }
        }
        /// <summary>
        /// 手机
        /// </summary>
        public string Mobilephone
        {
            get { return _mobilephone; }
            set
            {
                if( value!  = null && value.Length > 40)
                throw new ArgumentOutOfRangeException("
手机号值错误", value, value.ToString());
                _mobilephone = value;
            }
        }
        /// <summary>
        /// Email
        /// </summary>
        public string Email
        {
            get { return _email; }
            set
            {
                if( value!  = null && value.Length > 100)
                    throw new ArgumentOutOfRangeException("
Email 值错误", value, value.ToString());
                _email = value;
            }
        }
        /// <summary>
        /// 通信地址
        /// </summary>
        public string Address
        {
            get { return _address; }
            set
            {
                if( value!  = null && value.Length > 200)
               throw new ArgumentOutOfRangeException("通
信地址值错误", value, value.ToString());
                _address = value;
            }
        }
        /// <summary>
        /// 邮编
        /// </summary>
        public string Zipcode
        {
            get { return _zipcode; }
            set
            {
                if( value!  = null && value.Length > 20)
               throw new ArgumentOutOfRangeException ("
邮编值错误", value, value.ToString());
                _zipcode = value;
            }
        }
        /// <summary>
        /// 备注
        /// </summary>
        public string Remark
        {
            get { return _remark; }
            set
            {
                if( value!  = null && value.Length > 200)
          throw new ArgumentOutOfRangeException ("备
```

FOLLOW MASTER PROGRAM

```
注值错误", value, value.ToString());
                    _remark = value;
                }
            }
            /// <summary>
            /// 状态
            /// </summary>
            public string Status
            {
                get { return _status; }
                set
                {
                    if( value! = null && value.Length > 20)
                throw new ArgumentOutOfRangeException("状
态值错误", value, value.ToString());
                    _status = value;
                }
            }

            /// <summary>
            /// 注册时间
            /// </summary>
            public DateTime Registerdate
            {
                get { return _registerdate; }
                set {_registerdate = value; }
            }
            #endregion      定义公有属性
        }
    }
```

（1）添加，添加也即插入，使用的 SQL 语句是 insert。XML 数据映射配置信息如下：

```
    <insert id="InsertSysuser" parameterClass="Sysuser">
     INSERT INTO DEAN.SYSUSER
        (USERID,PASSWORD,LOGINNAME,SEX,BIRTHDAY,
IDCARD,OFFICEPHONE,FAMILYPHONE,MOBILEPHONE,
EMAIL,ADDRESS,ZIPCODE,REMARK,STATUS)
     VALUES    (#Userid#,#Password#,#Loginname#,#Sex#,
#Birthday#,#Idcard#,#Officephone#,#Familyphone#,
#Mobilephone#,#Email#,#Address#,#Zipcode#,#Remark#,
#Status#)
    </insert>
```

insert 标签表面该语句是插入语句，id 为该配置语句的标识，在后面的调用中通过该标识引用这个语句。变量用 # 号分隔，如“#Userid#”在运行期会由传入的 Sysuser 对象的 Userid 属性填充，注意变量名需区分大小写。调用代码如下：

```
        protect ed void BtnAddUser_Click (object sender,
EventArgs e)
        {
            try
            {
                iBatisTest.Domain.Sysuser model = new
iBatisTest.Domain.Sysuser();//实例化 Sysuser 对象
                model.Userid = 1;//Userid 为主键,多次添加需修改该值
                model.Password = "123";
                model.Loginname ="dean";
                model.Sex = "男";
                model.Birthday = Convert.ToDateTime("1980-01-01");
                model.Idcard ="510200198001014200";
                model.Officephone="86279528";
                model.Familyphone ="86279528";
                model.Mobilephone = "13880980000";
                model.Email ="476408321@qq.com";
                model.Address = "四川成都";
                model.Zipcode = null;
                model.Remark = null;
                model.Status = "有效";
                ISqlMapper mapper = Mapper.Instance();
//得到 ISqlMapper 实例
                mapper.Insert ("SysuserMap.InsertSysuser",
model);//调用 Insert 方法
                Label1.Text = "添加用户成功";
            }
            catch (Exception ex)
            {
                Label1.Text = ex.Message;
            }
        }
```

调用程序先实例化 Sysuser 对象并赋值，再得到 ISqlMapper 的实例，调用该实例的 Insert 方法。Insert 方法有两个参数，分别是语句名称和参数对象。语句名称就是数据映射文件里面的配置语句 id，参数对象是一个 object 对象，可以传入类、单个值或者哈希表等。

编译程序，点击“添加用户”按钮运行调用程序，系统会成功地向数据库 SYSUSER 表新增一条记录。查询数据库结果如图 2 所示。

图 2　新增数据后数据库结果

（2）更新，更新修改操作使用的 SQL 语句为 update，XML 数据映射文件配置信息为：

```
<update id="UpdateSysuser" parameterClass="Sysuser">
UPDATE SYSUSER
        SET  PASSWORD =#Password#,LOGINNAME =
#Loginname#,SEX =#Sex#,BIRTHDAY =#Birthday#,IDCARD =
#Idcard#,OFFICEPHONE =#Officephone#,FAMILYPHONE =
#Familyphone#,MOBILEPHONE =#Mobilephone#,EMAIL =
#Email#,ADDRESS=#Address#,ZIPCODE=#Zipcode#,REMA
RK=#Remark#,STATUS=#Status#
WHERE USERID = #Userid#
</update>
```

调用代码为：

```
protect ed void BtnUpdateUser_Click (object sender,
EventArgs e)
    {
        try
        {
     iBatisTest.Domain.Sysuser model = new iBatisTest.
Domain.Sysuser();
            model.Userid = 1;//修改 Userid 为 1 的用户信息
            model.Password = "1234";
            model.Loginname = "deanu";
            model.Sex = "男";
  model.Birthday = Convert.ToDateTime("1970-01-01");
            model.Idcard = "310200198001014200";
            model.Officephone = "76279528";
            model.Familyphone = "76279528";
            model.Mobilephone = "13880000000";
            model.Email = "1@qq.com";
            model.Address = null;
            model.Zipcode = "610000";
            model.Remark = "修改 ID 为 1 的用户";
            model.Status = "有效";
             ISqlMapper mapper = Mapper.Instance(); //得到
//ISqlMapper 实例
            mapper.Update ("SysuserMap.UpdateSysuser",
model);//调用 Update 方法
            Label1.Text = "修改用户成功";
        }
        catch (Exception ex)
        {
            Label1.Text = ex.Message;
        }
    }
```

（3）删除，删除操作使用的 SQL 语句为 delete，XML 数据映射文件配置信息为：

```
  <delete id="DeleteSysuser" parameterClass="int">
       DELETE FROM SYSUSER WHERE USERID =
#Userid#
  </delete>
```

由于删除操作是根据主键进行处理的，传入参数为整形值，因此在 parameterClass 直接为 int。调用代码为：

```
protected void BtnDelete_Click(object sender, EventArgs e)
    {
        //删除用户
        try
        {
            ISqlMapper mapper = Mapper.Instance(); //得到
//ISqlMapper 实例
        mapper.Delete("SysuserMap.DeleteSysuser", 1);
//调用 Delete 方法
            Label1.Text = "删除用户成功";
        }
        catch (Exception ex)
        {
            Label1.Text = ex.Message;
        }
    }
```

（4）查询，查询操作使用的语句为 select 语句，XML 数据映射文件配置信息为：

```
<select id="SelectSysuser" parameterClass="int" resultMap="
SysuserResult">
        SELECT  USERID,PASSWORD,LOGINNAME,SEX,
BIRTHDAY,IDCARD,OFFICEPHONE,FAMILYPHONE,
MOBILEPHONE,EMAIL,ADDRESS,ZIPCODE,REMARK,
STATUS,REGISTERDATE
  FROM SYSUSER WHERE USERID = #Userid#
</select>
```

调用代码为：

```
protected void BtnQuery_Click(object sender, EventArgs e)
    {
        //查询用户
        try
        {
            ISqlMapper mapper = Mapper.Instance(); //得到
//ISqlMapper 实例
            int Userid = 1;
             IList <iBatisTest.Domain.Sysuser > plist =
mapper.QueryForList<iBatisTest.Domain.Sysuser>("Sysuser
Map.SelectSysuser", Userid);//调用 QueryForList 方法
            if (plist ! = null && plist.Count > 0)
            {
                gvUserData.DataSource = plist;
                gvUserData.DataBind();
            }
            Label1.Text = "查询用户成功";
        }
        catch (Exception ex)
        {
            Label1.Text = ex.Message;
        }
    }
```

调用 ISqlMapper 实例的 QueryForList 方法返回的是一个

IList 泛型接口，这里指定为 iBatisTest.Domain.Sysuser 对象。程序运行结果如图 3 所示。

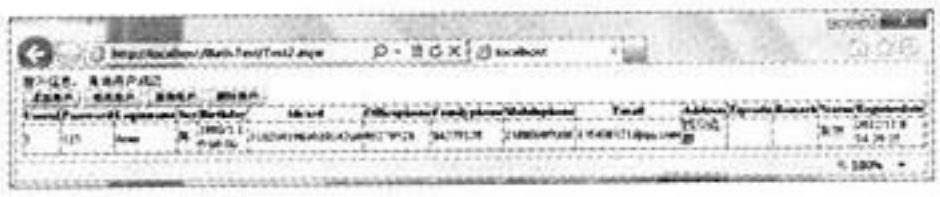


图 3　查询结果

以上 4 种配置语句都可以用 statement 来代替，其配置信息和调用代码完全一样。但用 statement 定义所有操作，缺乏直观性。建议在开发中根据实际情况来选用配置，既使得配置文件直观，又可以借助 xsd 对节点声明进行更有针对性的检查，以避免配置上的失误。

4　特殊配置

4.1　XML 转义字符

很多时候在 SQL 语句中会用到大于或者小于符号（即：><），这个时候就与 XML 规范相冲突，影响 XML 映射文件的合法性。通过加入 CDATA 节点来避免这种情况的发生，如：

```
<statement id="SelectPersonsByAge" parameterClass="int"
resultClass="Person">
  <! [CDATA[
    SELECT * FROM PERSON WHERE AGE > #value#
  ]]>
</statement>
```

4.2　自动生成主键

目前很多数据库都支持为新插入的记录自动生成主键，Oracle 也提供这种功能，通过序列加触发器来解决。这当然很方便,但是当希望在插入一条记录后立即获得该记录的主键值，问题就出现了，因为很可能为此不得不新写一条语句去获取该主键值。iBatis.Net 提供了一种比较好的解决方式。通过在 XML 数据映射文件中的<selectKey>元素来获取这些自动生成的主键值并将其保存在对象中。

如上面的“添加”操作配置信息修改为：

```
<insert id="InsertSysuser" parameterClass="Sysuser">
<selectKey resultClass="int32" property="Userid" type="pre">
    SELECT  DEAN.SEQ_SYSUSER_USERID.NEXTVAL  AS
Userid FROM DUAL
  </selectKey>
   INSERT  INTO  DEAN.SYSUSER  (USERID,PASSWORD,
LOGINNAME,SEX,BIRTHDAY,IDCARD,OFFICEPHONE,
FAMILYPHONE,MOBILEPHONE,EMAIL,ADDRESS,
ZIPCODE,REMARK,STATUS)
     VALUES     (#Userid#,#Password#,#Loginname#,#Sex#,
#Birthday#,#Idcard#,#Officephone#,#Familyphone#,
#Mobilephone#,#Email#,#Address#,#Zipcode#,#Remark#,
#Status#)
</insert>
```

粗体部分为新加的 selectKey 节点。#Userid# 参数将使用节点中从序列里选出的值进行填充。修改调用的代码：

```
ISqlMapper mapper = Mapper.Instance(); //得到
//ISqlMapper 实例
Int32 result =  (Int32)mapper.Insert ("SysuserMap.
InsertSysuser", model);//调用 Insert 方法
Label1.Text = " 添加用户成功,result 返回 UserID 为"
+ Convert.ToString (result) + ",Sysuser 实例 model 的 UserID
为" + Convert.ToString(model.Userid);
```

运行程序，这样 result 就得到想要获取的键值，同时 Sysuser 的实例化对象 model 的 UserID 属性也返回了该条新记录的键值。

4.3　# 与$的选择

在 XML 数据映射文件中，参数用 # 号来表示。如前面提到的 #Userid#，同时也可以使用$来标识。两者区别，以及如何选择介绍如下：

（1）# 是把传入的数据当作参数，如 order by #field#，如果#field# 传入的值是字符型，传入值为 id, 则 sql 语句会生成 order by " id"，很明显生成的 sql 语句提交给数据库执行会报错。# 参数往往用在需传入实际变量值的情况，如 where id= #id#，变量 #id# 传入 10，则 sql 语句会生成 where id=10。

（2）$采用拼接方式生成 sql 语句，即传入的数据直接生成在 sql 语句里，如 order by $field$，如$field$传入的是 id，则 sql 语句会生成 order by id。$参数方式一般用于传入数据库对象。例如传入表名。不过要特别提醒，因为使用该参数是直接组成 sql 语句，所以需注意防止 sql 注入。

4.4　LIKE 语句处理

在查询处理过程中，模糊查询经常会用到。iBatis 如何处理这种模糊查询呢？处理 Like 模糊查询 iBatis 提供以下两种方法：

（1）使用 # 参数： 配置如 Where UserName Like #param#‖'%', 采用参数传递的方式，如 #param# 传入字符 d，生成的 sql 语句为 Where UserName Like 'd%'，其他模糊情况同理可以相应配置出来。

（2）使用$参数：配置如 Where UserName Like '$param$%'，采用字符串替换，就是用参数的值替换 param，如$param$传入字符 d，也可以生成相同的语句。

$方式会引起 sql 注入风险，实际的使用中，建议大家都使用第一种配置方式来处理 LIKE 模糊查询。

5　结语

以上程序在 Windows 7（64 位）+VS2012+Oracle 11g（64 位）上测试通过。

（收稿日期：2012-11-09）

理论考核试卷自动生成

王文举 邢业伟 聂电开 葛 伟

摘 要：介绍一种随机抽取题库生成试卷的方法，并给出了核心算法和代码实现。

关键词：C# 语言；随机抽取；试卷生成

1 引言

理论考核通常要从题库中按照方案随机抽取题目生成试卷，下文介绍一种自动生成试卷的实现方法，主要包括设计思路和实现代码，设计思路主要介绍随机抽取题目的核心算法，实现代码包括算法的实现和试卷的生成，可供读者借鉴使用。

图 1 试卷抽取方案界面示例

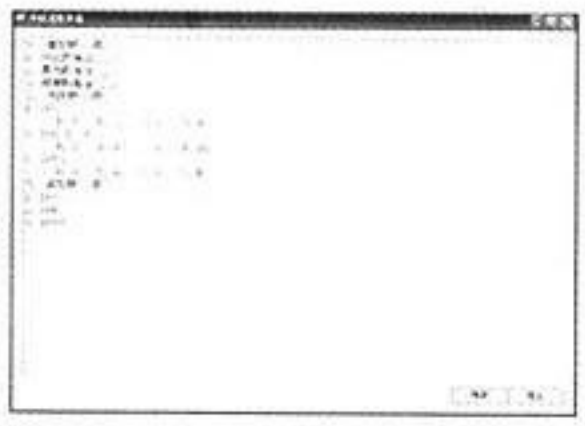


图 2 试卷随机生成界面示例

2 设计思路

数据库采用 Access，编程语言采用 C#。如图 1 所示，按照方案选取填空题、选择题、简答题的数量，点击“试卷生成”按键，进入图 2 界面，界面显示为随机生成的试卷，点击“保存”按键，保存为两个 Word 文档，一份是试卷，一份是试卷答案。

从题库中随机抽取题目的核心算法，其功能函数为 int [ ] GetRandoms (int maxValue, int count)，函数代码在下文。以填空题为例，如题库中有 N 道题目，需要抽取 M 个题目 (M<=N)。

（1）建立个长度为 N 的数组 int [ ] intList，保存 0 至 (N-1)，示意如图 3 所示。

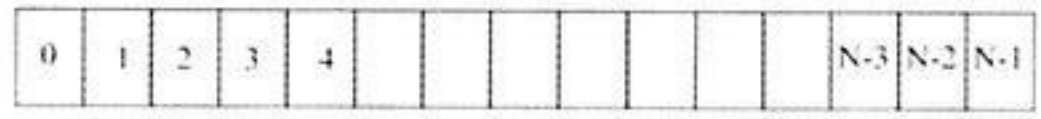


图 3 数组 intList 示例

建立个长度为 M 的数组 int [ ] intRet，示意如图 4 所示。

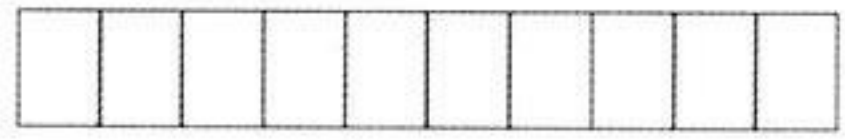


图 4 数组 intRet 示例

（2）针对数组 intList，从 [0,N-1) 中随机抽取一个整数 m，那么 0≤m≤N-2，取 intList [m] 值赋给 intRet [0]；并且取 intList [N-1] 值赋给 intList [m]。

（3）针对数组 intList，从 [0,N-2) 中随机抽取一个整数 m1，那么 0≤m1≤N-3，取 intList [m1] 值赋给 intRet [1]；并且取 intList [N-2] 值赋给 intList [m1]。

（4）同上方法循环 M 次，填充数组 intRet [ ]，其中的值为从 intList 中随机抽取的值。

（5）读取题库数据表，生成数组 DataRow [ ]，例如：

```
DataRow [] dRows = dataOper.dataSet.Tables [" TKT"] .
Select () ;
```

（6）以数组 intRet [ ] 中的值，依次取数组 DataRow [ ] 中的值，例如：

```
DataRow dr = dRows [intRet [i]] ;
```

上述方法，一方面很好地利用随机函数抽取数值，另一方面有效地避免了抽取数值有重复项，是一种非常灵巧的随机抽取算法，可以应用于其他随机抽取过程中。

3 实现代码

下面主要介绍图 2 的实现代码，包括界面生成函数，随机抽取题库，随机抽取函数，以及生成 Word 试卷和答案：

```
public partial class Sjsc : Form
{  //数据库操作的类
    DoDataOperator dataOper = new DoDataOperator();
```

```
    int tkt, xzt, jdt; //填空题,选择题,简答题数量
    //构造函数
        public TjBaogao(int _tkt, int _xzt, int _jdt)
    {   tkt = _tkt; xzt = _xzt; jdt = _jdt;
        InitializeComponent();
    }
     //界面生成函数
private void TjBaogao_Load(object sender, EventArgs e)
    {
        DataRow dr;
                    dataOper.ConnectionString = publicConn.
GetConnectionString();
            dataOper.InitStatus();
    string th = ""; string tm = ""; string da = ""; string xx = "";
            //读取数据库中数据表
    dataOper.AddTableFormSql(@"select * from tkt", "TKT");
DataRow[] dRows = dataOper.dataSet.Tables["TKT"].Select();
            if (dRows.Length >= tkt)
            {
                //随机抽取填空题
      int[] numbers = GetRandoms(dRows.Length, tkt);
       richTextBox1.Text += "一、填空题(" + tkt + "题)\n";
      richTextBox2.Text += "一、填空题(" + tkt + "题)答案\n";
                for (int i = 0; i < numbers.Length; i++)
                {
                    dr = dRows[numbers[i]];
                th = (i + 1).ToString();
                tm = Convert.ToString(dr["zb_tm"]);
                da = Convert.ToString(dr["zb_da"]);
                        //题号,题目写入 richTextBox1
                richTextBox1.Text += th + ". " + tm + "\n";
                        //题号,答案写入 richTextBox2
                richTextBox2.Text += th + ". " + da + "\n";
            }
        }
        else
        {
         publicConn.MessageText("题库中填空题的数量小于
抽取的数量,请维护题库或减少抽取数量! ");
         return;
        }
        //选择题,简答题实现方法同上
    }
      /// <summary>
      /// 随机抽取函数,从 maxValue 中随机抽取 count 个数,
/// 返回数组
      /// </summary>
          /// <param name="maxValue">题库中数量</param>
           /// <param name="count">考试中需要抽取的数量</
/// param>
      /// <returns></returns>
      private int[] GetRandoms(int maxValue, int count)
{
    int[] intList = new int[maxValue];
    for (int i = 0; i < maxValue; i++)
    { intList[i] = i;}
    int[] intRet = new int[count];
    int n = maxValue;
    Random rand = new Random();
    for (int i = 0; i < count; i++)
    {
        int index = rand.Next(0, n);
        intRet[i] = intList[index];
        intList[index] = intList[--n];
    }
    return intRet;
}
    //生成 Word 试卷和答案
private void buttonSave_Click(object sender, EventArgs e)
    {
    saveFileDialog1.Filter = "Word files (*.doc)|*.doc";
    DialogResult dr = saveFileDialog1.ShowDialog();
    if (dr == DialogResult.Cancel)
    { return; }
    string fn = saveFileDialog1.FileName;
    if (File.Exists(fn))
    { File.Delete(fn);}
        richTextBox1.SaveFile(fn); //题号,题目写入名称为
//"fn"的 Word 文档
  //设置答案文件名,答案文件名比考题文件名多"答案"两个字
        int i = 0; int start = 0;
        foreach (char a in fn)
        {
            if (a == '\\')
            { start = i; }
            i++;
        }
         string fn2 = fn.Substring (0, start) + fn.Substring
(start, fn.Length - 4 - start) + "答案" + ".doc";
        if (File.Exists(fn2))
        { File.Delete(fn2);}
          richTextBox2.SaveFile(fn2); //题号,答案写入名称为
//"fn 答案"的 Word 文档
        publicConn.MessageText("保存成功! ");
    }
}
```

4 结语

主要介绍从题库中随机抽取题目生成试卷的方法，并且详细介绍了随机抽取的核心算法，给出了代码实现。该方法非常灵巧，可供读者灵活应用于多种随机抽取过程中。

（收稿日期：2012-09-15）

理解 C# 中的委托和事件

黎 明

摘 要：用简明的例子说明了 C# 中的委托和事件，帮助读者深入理解，同时也有助于读者对 C# 的后继学习。

关键词：C# 语言；委托；事件

委托是 C# 中比较重要的概念，学习 C# 在这里最容易产生迷惑，理解后对后面内容的学习很有帮助。

有些时候，由于在开发程序时对后续可能出现的要求及变化考虑不足而导致一些麻烦，这些新变化可能会导致程序的重新编写，那能不能改变这种情况？后面的需求变化了，怎样使后续对应功能的编写对前面的程序不造成影响？在 C# 中可以用委托来解决这个问题。举个简单的例子。

比如一个数据表需要导出，在开始只是设计了导出到 TXT 和 Excel，程序如下：

```
//定义类及使用方法
class HrDataInfo
{
    //定义委托
    public delegate void SaveAsDelegate(string FileName);
    public SaveAsDelegate SaveAs;

    public void SaveAsTxt(string FileName)
    {
        Console.WriteLine (" 将数据导出到 TXT 文件！ {0}",
FileName);
    }

    public void SaveAsExcel(string FileName)
    {
        Console.WriteLine (" 将数据导出到 Excel 文件！ {0}",
FileName);
    }
}
```

上面定义了一个 HrDataInfo 类，类中有两个方法，一个是 SaveAsTxt (string FileName)，一个是 SaveAsExcel (string FileName)，如何让用户来决定使用哪一个呢？

玄机就在于上面的两行代码：

```
public delegate void SaveAsDelegate(string FileName);
public SaveAsDelegate SaveAs;
```

定义一个委托，先保证和定义方法一样，等于定义一个统一的方法模子 SaveAsDelegate (string FileName)，委托实质上是类，因为不是静态的所以要实例化为 SaveAs。

```
//使用
class Program
{
 static void Main(string[] args)
 {
    //实例化类
    HrDataInfo Hr = new HrDataInfo();

    //输出到 Excel 文件
    Hr.SaveAs = Hr.SaveAsExcel;
    Hr.SaveAs(@"C:\1.xls");

    //输出到 TXT 文件
    Hr.SaveAs = Hr.SaveAsTxt;
    Hr.SaveAs(@"C:\1.TXT");

    //输出到 Word 文件，是静态方法，直接使用
    Hr.SaveAs = SaveAsWord;
    Hr.SaveAs(@"C:\1.Word");

    //输出到 PPT 文件，非静态，实例化对象后引用
    OthersDealWith ODW=new OthersDealWith ();
    Hr.SaveAs =ODW.SaveAsPPT;
    Hr.SaveAs(@"C:\1.PPT");

    Console.ReadKey();
 }
 static void SaveAsWord(string FileName)
 {
     Console.WriteLine (" 将数据导出到 Word 文件！ {0}",
FileName);
 }
 }
 public class OthersDealWith
 {
  public void SaveAsPPT(string FileName)
  {
```

```
        Console.WriteLine (" 将数据导出到 PPT 文件！{0}",
FileName);
    }
    }
```

因为原来的类中只有导出到 Excel 和 TXT，现在需要导出到 Word 和 PPT，而原来的类中没有这个方法。

需要写上相应的处理方法，然后赋值给委托。

```
//另外的写法
 Hr.SaveAs = new HrDataInfo.SaveAsDelegate (Hr.
SaveAsExcel);
    Hr.SaveAs(@"C:\1.xls");
```

也是对的，是一般的写法。

总结：

(1) 委托实际上就是指针，就是方法的地址，程序中让它指向哪个方法它就指向哪个方法；

(2) 委托是统一的方法模型，参数必须一致。

(3) 委托实际上是把方法当作参数来传递，可以是静态方法也可以是非静态的方法。

从上面的例子中看出，类中定义了委托给程序带来了很大的灵活性，有一个类放在那里，里面藏了个指针，让它指向哪里它就指向哪里（当然有约定）。这让我们想到了事件，比如一个按钮放在窗体上，如果里面也藏了这么个东西，是不是就可以处理相应的点击，比如，点击了按钮，我让它指向一个处理程序，那么这个是不是就有了所谓的按钮响应，其实这就是事件，叫按钮的点击事件。

C# 中是不是这样处理的呢？

新键一个窗体程序，随便放一个按钮 button1 到窗体 form1 上，在没有为按钮写处理程序（鼠标点击）之前，Form1.Designer.cs 中的代码是这样的：

```
 this.button1.Location = new System.Drawing.Point(29, 51);
    this.button1.Name = "button1";
    this.button1.Size = new System.Drawing.Size(75, 23);
    this.button1.TabIndex = 0;
    this.button1.Text = "button1";
    this.button1.UseVisualStyleBackColor = true;
```

现在，双击按钮 button1，为它编写鼠标点击事件，看看系统做了什么？

第一个变化，系统自动加了如下的代码：

```
    private void button1_Click(object sender, EventArgs e)
    {
    }
```

第二个变化，还是看 Form1.Designer.cs，

```
this.button1.Location = new System.Drawing.Point(29, 51);
    this.button1.Name = "button1";
    this.button1.Size = new System.Drawing.Size(75, 23);
    this.button1.TabIndex = 0;
    this.button1.Text = "button1";
    this.button1.UseVisualStyleBackColor = true;
     this.button1.Click += new System.EventHandler (this.
button1_Click);
```

对比上面的代码，多了：

```
    this.button1.Click += new System.EventHandler (this.
button1_Click);
```

(+=,-=,+,-,是委托的运算,+和+=是订阅,-和-=是退订)

原来就是这样的：this.button1.Click 就是个委托，只是系统自动地把它指向了 button1_Click (object sender, EventArgs e)，所以，用户在按钮 Button 上点鼠标系统就自动运行 button1_Click (object sender, EventArgs e) 里面的程序。

也可以改动这些系统自动生成的代码（一般建议不要动）。比如，写如下程序：

```
    private void MineBT1_Click(object sender, EventArgs e)
    {
        MessageBox.Show("你点击了按钮！");
    }
```

然后改动 Form1.Designer.cs 中的代码。把：

```
 this.button1.Click += new System.EventHandler (this.
button1_Click);
```

替换为：

```
this.button1.Click+=this.MineBT1_Click;
```

运行程序，一样得到的结果。

另外，在 WPF 程序中，还可以通过属性赋值的方式来把事件处理程序与事件的拥有者联系在一起，比如在窗体上放置一个按钮，只要为这个按钮的单击事件写了响应程序，系统自动把二者结合起来 [绑定]，就是属性赋值。

```
Click="button1_Click"
```

当然也可以在程序中删除它，通过 C# 语言来处理。

在类中写下：

```
this.button1.Click+=new RouteDEventHandler(button1_Click);
```

效果和属性赋值是一样的。

这自定义控件中，如何定义事件响应，比如用户做了个自定义控件，里面有个按钮，它想让用户点击这个按钮时运行用户所写的代码，因为控件已经封装好了，里面定义好按钮的事件，在控件里面或者外面写处理方法，按照上面的模式处理就 OK 了。

（收稿日期：2013-01-11）

Delphi 开发 SAP/R3 系统的外围接口程序

郭海伟 李青龙

摘 要：结合在项目开发中的实际经验，通过一个具体的示例，阐述通过 RFC（远程调用）的方式，使用 Delphi 开发 SAP/R3 系统的外围接口应用程序。

关键词：Delphi 语言；SAP 系统；ERP 软件

1 引言

SAP (Systems, Application and Products in Data Processing) 是 ERP (Enterprise-wide Resource Planning) 解决方案的先驱，它为各种行业、不同规模的企业提供全面的解决方案。SAP 既是公司名称，也是其 ERP 的软件名称。SAP 公司成立于 1972 年，总部位于德国，是全球最大的企业管理和协同化电子商务解决方案供应商，世界 500 强中 80%以上的公司都在使用 SAP 的管理解决方案，近年来国内 90%的大型国企、民企也都成功实施了 SAP 系统。但是这样问题就出现了，随着 SAP 系统的实施，企业的信息化水平虽然得到了提高，但企业已经经过多年的信息化建设，已经为不同部门、不同的应用建立了多个应用系统。由于这些信息系统都是为了不同的需要分别设计开发的，SAP 系统与这些系统之间不能有效共享业务处理流程和信息，不能有效地协同工作。企业迫切需要消除 SAP 系统与其他系统之间一个个的信息孤岛，使各个应用间既相互独立又能有效协同工作，将不同的应用集成到一个完整的企业级信息化环境中。SAP 接口技术很好地解决了 SAP 系统与外围信息系统的集成问题以及数据迁移或共享问题。

2 实施细则

SAP R/3 的接口方式主要有 RFC、IDOC、BAPI 3 种，下文要介绍的是相对比较简单的 RFC（Romote Function Call，远程函数调用）。SAP 系统 RFC 调用的原理其实很简单，有一些类似于三层构架的 C/S 系统，第三方的客户程序通过接口调用 SAP 内部的标准或自定义函数，获得函数返回的数据进行处理后显示或打印。图 1 是 RFC 调用的模型。SAP R/3 的接口方式主要有 RFC、IDOC、BAPI 3 种，文中介绍的是相对比较简单的 RFC（Romote Function Call，远程函数调用）。SAP 的 RFC 调用是其接口技术中最简单和易用的一种方式，该方式开发比较简便，特别适合于外部报表开发，主要介绍采用 Delphi7 开发采用 RFC 方式与 SAP 应用系统交互的接口应用程序。

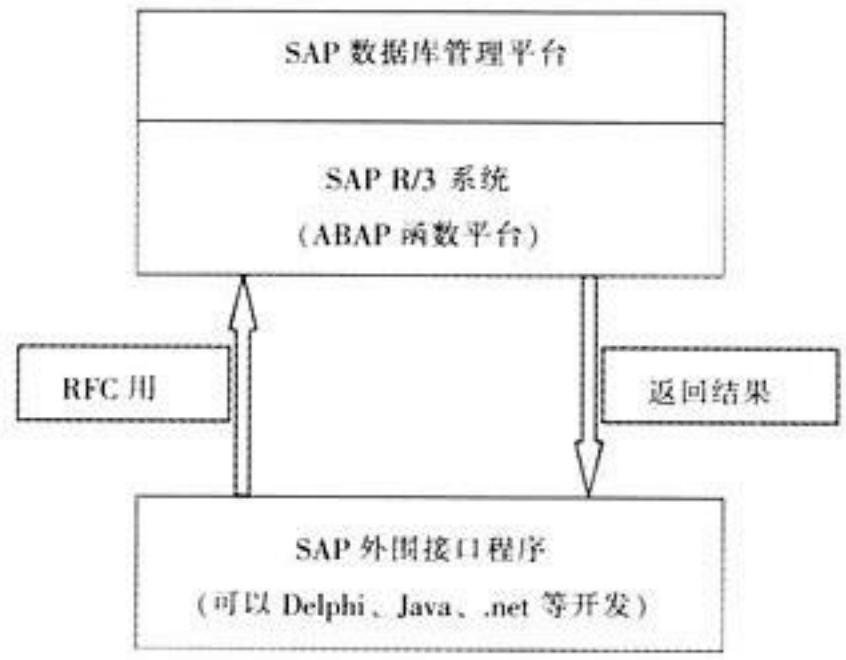


图 1 RFC 调用模型

第一步是建立 Delphi 接口应用程序与 SAP 应用系统间的连接。

首选在 Delphi 系统中安装 SAP 的 SDK 控件：SAPLogonControl，SAPBapiControl。

在 Delphi7 菜单中选择 Component →Import ActiveX Control (TSAPLogonControl, TSAPBapiControl)，如图 2 所示。

图 2 安装控件

做完这一步骤后，可以在 Delphi 中看到控件如图 3 所示。

图 3 控件安装成功

接下来开始写 Delphi 代码，测试连接。

放两个按钮：一个是建立 SAP 连接，一个是调用 SAP FUNCTION，如图 4 所示。以下是建立连接的代码：

图 4　添加按钮

```
procedure TForm1.btn_ConnAspClick(Sender: TObject);
Begin
//以下服务器参数请根据客户配置情况更改
    Connection := saplogoncontrol1.NewConnection;
    Connection.User := Ansiuppercase('useName');
//SAP 登录用户名
    Connection.System  := 'IDS';
    Connection.Client  := '900'; 生产系统集团代码
    Connection.ApplicationServer := '10.75.3.20';  //生产系统
//服务器 IP
    Connection.SystemNumber := '00';  //生产系统号
    Connection.Password := 'password1';//SAP 登录用户密码
    Connection.Language := 'ZH' ; //SAP 系统界面中文
    //Connection.Codepage := '8000' ;
    SAPLogonControl1.Enabled := false;

    if Connection.LogOn(0,true) = True then
    begin
      ShowMessage('Logon O.K.');
      SapBapiControl1.Connection:=Connection;
      sapFunctions1.Connection := Connection;
    end
    else
    begin
      ShowMessage('Error on logon :-(((');
    end;
end;
```

第二步是调用 SAP FUNCTION：BAPI_PO_CREATE。

```
procedure TForm1.btn_SelectFieldClick(Sender: TObject);
var
  txt,txt2:string;
  n,r,k,i:integer;
  head1 : Variant;
  time1 : TDateTime;
begin
    SAPFunctions1.RemoveAll;
     funct :=SAPFunctions1.add('RFC_READ_TABLE'); //通过
//RFC 接口远程运行 SAP 内部函数名
       funct.exports('QUERY_TABLE').VALUE:= Edit1.Text ;
//向函数入口赋查询表名称
    funct.exports('rowcount').VALUE:= '20';  //向函数入口赋值
    funct.exports('NO_DATA').VALUE:=  '1';
    IF NOT funct.CALL THEN
      ShowMessage(Funct.exception)
    ELSE
    BEGIN
      //TABLE:=funct.TABLES.ITEM('DATA');
      clear_myList;
      TABLE:=funct.TABLES.ITEM('FIELDS');
      //ShowMessage( IntToStr( TABLE.ROWCOUNT ));
      lbl_fieldcount.Caption := '字段总数:'+IntToStr ( TABLE.
ROWCOUNT );
      StringGrid1 .RowCount :=TABLE.ROWCOUNT + 1 ;
      StringGrid1 .Cells [0,0]:= '序号';
      StringGrid1 .Cells [1,0]:= '字段名';
      StringGrid1 .Cells [2,0]:= '工作区内域偏移';
      StringGrid1 .Cells [3,0]:= '长度(字符数)';
      StringGrid1 .Cells [4,0]:= 'ABAP 数据类';
      StringGrid1 .Cells [5,0]:= '资源库对象的短文本';
      CheckListBox1.Items.Clear;
      for r:=1 to StringGrid1 .RowCount –1 do
      begin
        StringGrid1 .Cells [0,r]:=IntToStr(r);
        StringGrid1 .Cells [1,r]:=Table.value(r,1);
        StringGrid1 .Cells [2,r]:=Table.value(r,2);
        StringGrid1 .Cells [3,r]:=Table.value(r,3);
        StringGrid1 .Cells [4,r]:=Table.value(r,4);
        StringGrid1 .Cells [5,r]:=Table.value(r,5);
        CheckListBox1.Items.Add(Table.value(r,1)+'  '+Table.
value(r,2) +'  '+Table.value(r,3)+'  '+Table.value(r,4)+'  '+
Table.value(r,5) +' | '+inttostr(r) );
        New(ARecord);
        ARecord^.FIELDNAME := Table.value(r,1);
        ARecord^.OFFSET    := Table.value(r,2);
        ARecord^.LENGTH    := Table.value(r,3);
        ARecord^.TYPE1     := Table.value(r,4);
        ARecord^.FIELDTEXT := Table.value(r,5);
        MyList.Add(ARecord);
        if (ARecord^.OFFSET+ARecord^.LENGTH +r)<512 then
          CheckListBox1.Checked [CheckListBox1.Items.Count
–1] := True;
      end;
    end;
    CheckListBox1ClickCheck(nil);
end;
```

（下转第 26 页）

用 VC++ 实现 RTSS 与 Win32 共享内存通信

赵常寿 吴红权 张 鹏

摘 要：利用共享内存对象实现 RTX 程序与 Win32 程序的进程间通信，可用于 RTX 实时系统软件开发中的数据传输。

关键词：共享内存； RTSS 实时子系统； Win32 程序； 进程间通信

1 引言

随着实时嵌入式技术的发展，Windows 系列操作系统越来越多地被考虑作为实时系统的应用平台。为了满足硬实时系统严格的响应时间要求，增加 Windows 系统的实时能力非常必要。美国 Ardence 公司的 RTX 产品，在 Windows 平台上提供了一个实时子系统（RTSS），实现了确定性的实时线程调度、实时环境和与原始 Windows 环境之间的进程间通信机制以及其他只在特定的实时操作系统中才有的对 Windows 系统的扩展特性，RTX 架构如图 1 所示。

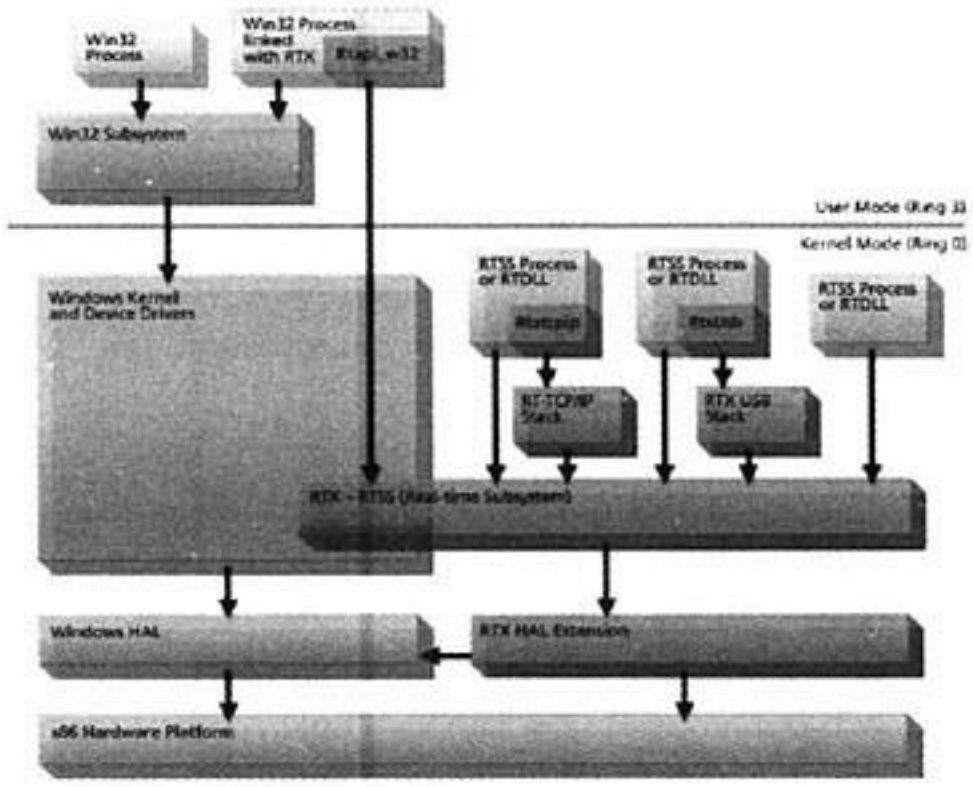


图 1 RTX 架构

基于 Windows XP SP3、RTX8.1.2 平台用 VC++2003 实现共享内存方式的进程间通信（IPC）。

2 软件结构

软件包含两个程序，一个是 Win32 程序，另一个是 RTSS 程序。Win32 程序就是普通的 Windows 应用程序，运行在 Win32 子系统下，实现界面和实时性要求不高的任务，RTSS 程序运行在 RTSS 实时子系统下，完成实时性要求高的任务，软件整体功能框架如图 2 所示。

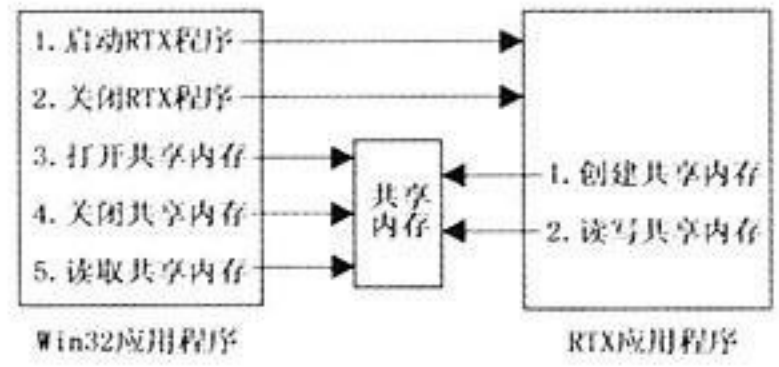


图 2 软件功能框架

软件运行界面如图 3 和图 4 所示。

图 3 RTSS 程序界面

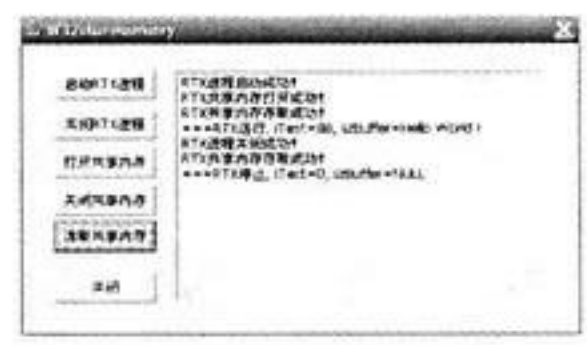


图 4 Win32 程序界面

3 功能实现

3.1 创建共享内存

在 RTSS 程序里创建一个命名的共享内存对象，用来与 Win32 程序传递信息，需要用到 RTAPI 函数 RtCreateSharedMemory，函数原型为：

```
HANDLE RtCreateSharedMemory(
    DWORD flProtect,
    DWORD MaximumSizeHigh,
    DWORD MaximumSizeLow,
    LPCTSTR lpName,
    VOID ** location
);
```

其中第一个参数是共享内存的保护模式，可以是 PAGE_READONLY 或 PAGE_READWRITE，第二个参数是共享内存大小的高32 位，第三个参数是共享内存大小的低 32 位，第四个参数是共享内存对象的名字，第五个参数是存储共享内存虚拟地址的指针。如果函数执行成功，返回值为共享内存对象句柄。程序代码如下：

```
if(! (hSM = RtCreateSharedMemory((DWORD)
PAGE_READWRITE,
                                        (DWORD)0,
    (DWORD)(sizeof(_sharedmemory)),
        chName,
            (void **)&pSM)))
{
    printf("====Create Shared Memory Error\n");
}
else
{
    printf("====Create Shared Memory OK\n");
    pSM->bStop = FALSE;
    pSM->iTest = 88;
    strcpy(pSM->szBuffer, "Hello World ! ");
    while(pSM->bStop == FALSE)
        RtSleep(10);

    pSM->iTest =0;
    strcpy(pSM->szBuffer, "NULL");
}
```

3.2 打开共享内存

在 Win32 程序里打开上面 RTSS 程序创建的共享内存对象，需要用到 RTAPI 函数 RtOpenSharedMemory，函数原型为：

```
HANDLE RtOpenSharedMemory(
    DWORD DesiredAccess,
    BOOL bInheritHandle,
    LPCTSTR lpName,
    VOID ** location
    );
```

其中第一个参数是访问模式，可以是 SHM_MAP_WRITE 或 SHM_MAP_READ，第二个参数忽略，第三个参数是共享内存对象的名字，第四个参数是存储共享内存虚拟地址的指针。如果函数执行成功，返回值为共享内存对象句柄。程序代码如下：

```
if ((hSM = RtOpenSharedMemory (SHM_MAP_WRITE,
FALSE, chName, (VOID **)&pSM)) == NULL)
{
    m_info.SetSel (m_info.GetWindowTextLength (),m_info.
GetWindowTextLength());
    m_info.ReplaceSel("RTX 共享内存打开失败！\n");
}
else
{
    m_info.SetSel (m_info.GetWindowTextLength (),m_info.
GetWindowTextLength());
    m_info.ReplaceSel("RTX 共享内存打开成功！\n");
}
```

3.3 启动 RTX 程序

一个 RTSS 进程可能通过以下方法中的任意一个来启动：

(1) 在系统启动时像加载驱动一样加载它（使用 RTSSrun /b）。

(2) 从命令行中运行 RTSS 可执行文件。

(3) 从 Win32 程序中启动 RTSS 进程。

从 Win32 程序中启动 RTSS 进程需要用到 Win32 API 函数 CreateProcess，函数原型为：

```
BOOL CreateProcess(
                    LPCTSTR lpApplicationName,
                    LPTSTR lpCommandLine,
        LPSECURITY_ATTRIBUTES lpProcessAttributes,
        LPSECURITY_ATTRIBUTES lpThreadAttributes,
                    BOOL bInheritHandles,
                    DWORD dwCreationFlags,
                    LPVOID lpEnvironment,
        LPCTSTR lpCurrentDirectory,
        LPSTARTUPINFO lpStartupInfo,
    LPPROCESS_INFORMATION lpProcessInformation
    );
```

具体参数说明可以参考 MSDN 文档。程序代码如下：

```
char chCMD[100]="RTSSRUN RTXsharememory.rtss";
if (CreateProcess(NULL,// application name -- not
// needed
        chCMD,// dos command line
        NULL,// process attributes
    NULL,// thread attributes
        FALSE,// handle inheritance flag
        DETACHED_PROCESS,// creation flags
        NULL,// pointer to new env. block
        NULL,// pointer to curr directory name
                &sinfo,// startup info
                &pinfo) == FALSE)// process info
{
    m_info.SetSel (m_info.GetWindowTextLength (),m_info.
GetWindowTextLength());
    m_info.ReplaceSel("RTX 进程启动失败！\n");
}
else
{
    m_info.SetSel (m_info.GetWindowTextLength (),m_info.
GetWindowTextLength());
    m_info.ReplaceSel("RTX 进程启动成功！\n");
    bRTX=TRUE;
    GetExitCodeProcess (pinfo.hProcess, &exitCode);
    CloseHandle(pinfo.hThread);
    CloseHandle(pinfo.hProcess);
}
```

4 结语

通过建立共享内存对象实现了在 RTSS 进程和 Win32 进程之间传递数据，可用于 RTX 实时系统软件开发中的数据传输。

（收稿日期：2013-01-28）

一种音频指纹构建与搜索架构的实现

明廷堂

摘　要：音频指纹技术是音频处理领域中的一种重要技术。它通过将一系列短促、无标记的音频片段与之对应的内容数据链接起来，为每个音频信号编制特征码，进而实现音频资源的匹配和识别。提出了一种音频指纹构建与搜索的基本架构，并采用 C# 编程语言详细实现了这一架构。

关键词：音频识别；音频指纹；签名

1　引言

多年来，研究者对音频识别进行了深入研究与分析。引入了计算机科学领域中的一个复杂问题：模拟信号数字化高效比较与识别技术。例如，假设现有两个音频信号 P_1 和 P_2，某个应用场景需要比较并确认它们是否自同一音源。在处理这种问题时，计算机系统不同于人类感官，首先需要对它们进行特定的数字化处理，然后才能按比特逐一比较处理后的二进制信号。同一信号的不同编码格式的各种内部特征 (比特率、采样率等) 使得它们产生差异。即使将音频文件转换成预定义的频谱，依然会因为误差、噪声等而得到不同的二进制表征。此时，音频指纹在这种应用场景中的优势就凸显了出来。

音频指纹是音频对象简短的摘要。它能将一系列短促、无标记的音频片断与之对应的内容数据链接起来。在对指纹 F 和音频对象 X 建立映射关系后，在进行音频指纹比对时，只须在两个相似的音频对象之间建立一种感知平等的有效的机制，不通过比较容量较大的音频对象本身，而是通过比较相关的指纹来完成。因而在音频处理领域得到广泛的应用：

(1) 歌曲的元数据的自动填充。

(2) 识别当前播放音乐。

(3) 监控广播电台。

(4) 解决 P2P 版权问题。

(5) 管理音频效果库。

音频指纹具备以下特性：

(1) 感知相关性。指纹中应尽可能多地包含感知相关的数据，而感知不相关的数据应尽可能从指纹中清除。

(2) 高效性。指纹应相对较小，以实现高效的检索。

(3) 鲁棒性。相似音频的指纹应该尽可能的一致，应该尽可能的抗击各种噪声。

目标是开发一个高效的音频指纹处理方法，用于音频指纹与信号识别。首先提出该算法架构，然后使用 C# 编程语言完整实现了该架构。另外，也涵盖了与该算法相关的数字信号处理领域的一些核心概念。

2　音频指纹构建与检索架构

一般而言，音频指纹算法架构可以分解为两个主要的逻辑部分：指纹创建 (从音频信号中提取的唯一感性特征)；指纹查找 (指纹库查询)。图 1 抽象的描述了音频指纹创建与检索算法的总体架构。

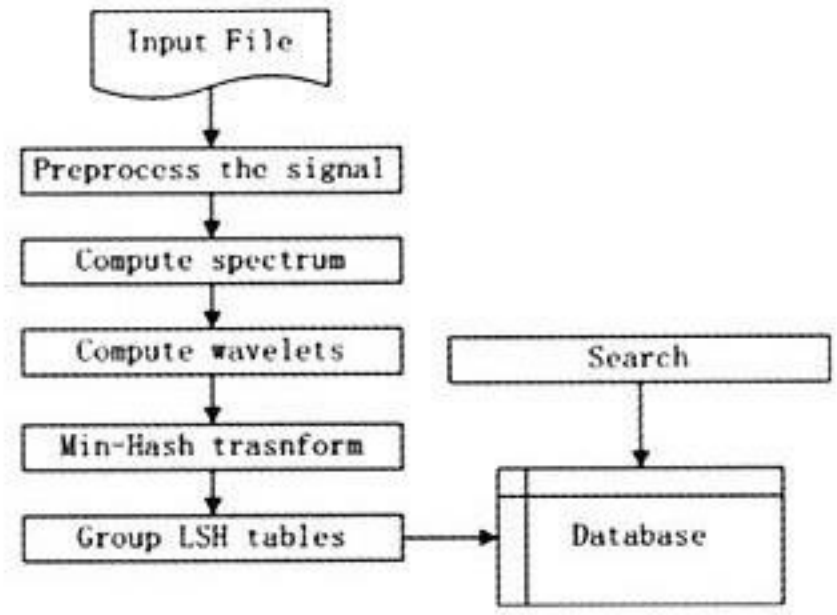


图 1　算法的总体架构

该算法包括下面一系列活动：

(1) 预处理信号，包括声轨转换、降频等操作。例如，对于输入的立体声文件 (stereo/44100Hz)，需要将其转换成单声轨、低频的信号 (mono/5512Hz)。

(2) 根据预处理的信号构建相应的频谱。

(3) 频谱图对数化，将其分割成子谱图。

(4) 使用 Harr 小波分解方法转换每个频帧。

(5) 每个音频指纹被编码成一个长度为 8192 的比特向量。

(6) 基于比特指纹使用 Min-Hash 散列方法产生相应的签名。

(7) 使用 LSH 技术将 25 个 Min-Hash 签名扩展成 25 个哈希表，用于音频指纹存查找。

(8) 音频指纹匹配与识别。

如图 2 所示的流程图重点描述了音频指纹构建部分的活动。

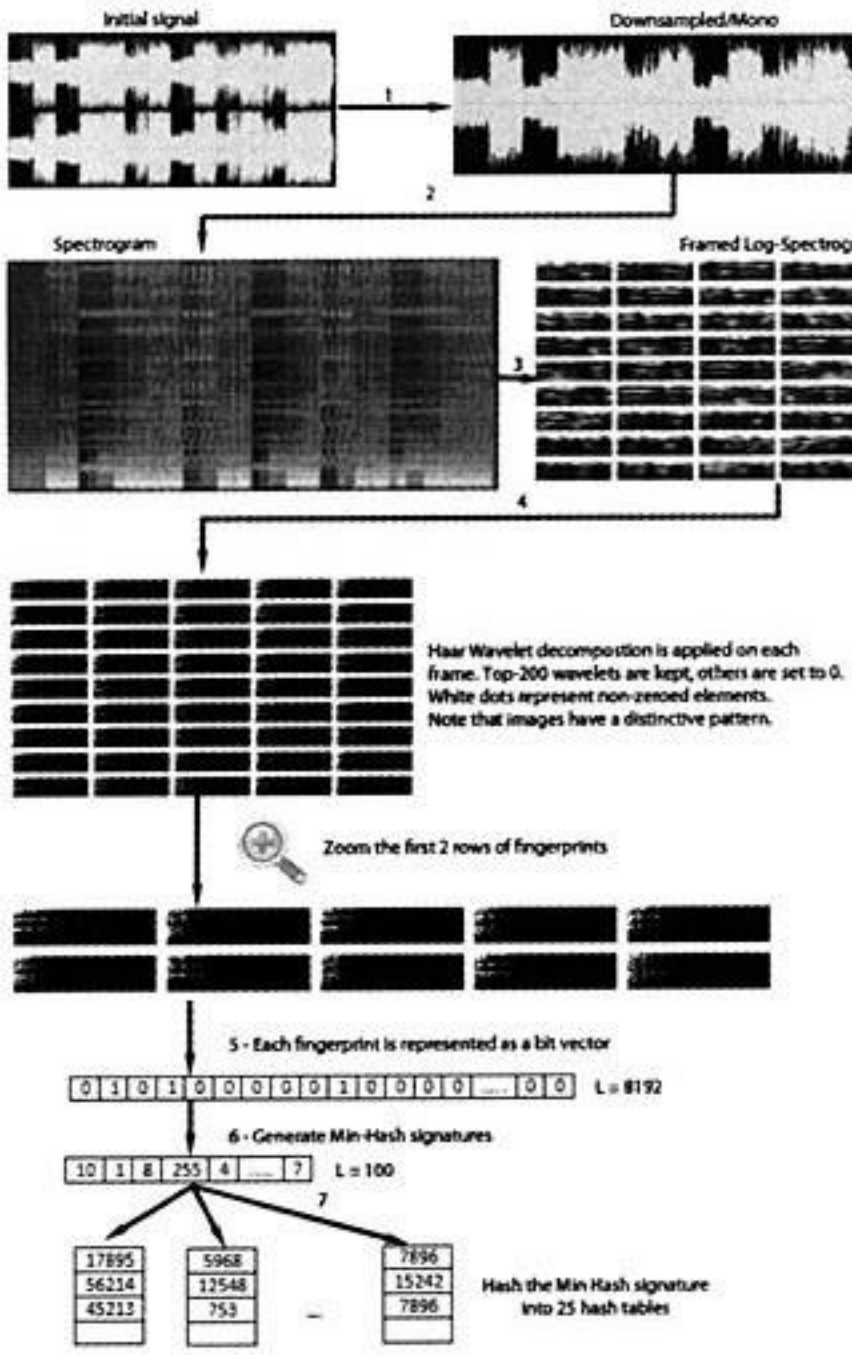


图 2　算法的活动图

3　架构实现

3.1　输入信号预处理

计算机上的所有音频文件都以特定的格式进行编码，音频指纹算法的第一步是对输入的音频文件解码成 PCM 格式，PCM 格式可被看作一个模拟信号的原始数字格式。在 PCM 中，信号幅度以均匀的间隔采样，每个采样都被量化为一系列数字增益中最接近的部分。紧随解码之后的是转换单声轨以及降低采样率。采样率定义了每秒从一个连续信号采集到的离散信号样本的数量。采样频率的倒数即为采样间隔或采样周期。虽然算法分析的频带范围是 318Hz~2000Hz，但降低采样频率到 5512Hz 通常是一个安全并且必要的操作。在降低采样率时，算法将丢弃与人类感知无关的信息，而关注信号最重要的特征。该预处理过程使用 Bass.Net 库来完成。

在检索数据时，主要收集-1.0~+1.0 范围的 32 比特浮点数样本。可以通过 Bass_ChannelGetData 函数检索样本数据。注意：任何压缩文件 (比如，MP3、OGG、WMA 等) 将被最先解码，使得开发者总是可以接收 PCM 采样，这主要得益于 Bass.Net 库的优势。另外，一个样本包含表征所有信道 (立体/单声) 上的离散信号所需的字节。

理论上，香农采样定理表明：当采样率大于信号被采样的最大频率的两倍时 (奈奎斯特频率超过信号被采样时的带宽)，完全重构一个信号是可能的。如果使用较低的采样率，采样信号将不能完全覆盖原始信号的信息。这也是标准的音频 CD 为何使用 44100Hz 的采样率，因为人耳通常只能感知频率大于 20000Hz 的声音。

下面是使用 Bass.Net 库进行解码的源代码片段：

```
/// <summary>
/// Read mono from file
/// </summary>
public float [] ReadMonoFromFile (string filename, int
samplerate,
            int milliseconds, int startmillisecond)
{
   int totalmilliseconds =
      milliseconds <= 0 ? Int32.MaxValue : milliseconds +
startmillisecond;
   float[] data = null;
   //create streams for re-sampling
   int stream = Bass.BASS_StreamCreateFile(filename, 0, 0,
        BASSFlag.BASS_STREAM_DECODE   |   BASSFlag.
BASS_SAMPLE_MONO |
        BASSFlag.BASS_SAMPLE_FLOAT); //Decode the
//stream
   if (stream == 0)
throw new Exception(Bass.BASS_ErrorGetCode().ToString());
    int mixerStream = BassMix.BASS_Mixer_StreamCreate
(samplerate, 1,
        BASSFlag.BASS_STREAM_DECODE   |   BASSFlag.
BASS_SAMPLE_MONO |
      BASSFlag.BASS_SAMPLE_FLOAT);
  if (mixerStream == 0)
 throw new Exception(Bass.BASS_ErrorGetCode().ToString());
                if (BassMix.BASS_Mixer_StreamAddChannel
(mixerStream, stream, BASSFlag.BASS_MIXER_FILTER))
   {
       int bufferSize = samplerate * 10 * 4; /*read 10
seconds at each iteration*/
      float[] buffer = new float[bufferSize];
      List<float[]> chunks = new List<float[]>();
      int size = 0;
     while ((float) (size)/samplerate*1000 < totalmilliseconds)
      {
         //get re-sampled/mono data
               int bytesRead = Bass.BASS_ChannelGetData
(mixerStream, buffer, bufferSize);
         if (bytesRead == 0)
            break;
          float[] chunk = new float[bytesRead/4]; //each float
//contains 4 bytes
         Array.Copy(buffer, chunk, bytesRead/4);
```

```
            chunks.Add(chunk);
            size += bytesRead/4; //size of the data
        }
            if ((float) (size)/samplerate*1000 < (milliseconds +
startmillisecond))
            return null; /*not enough samples to return the
requested data*/
 int start = (int) ((float) startmillisecond*samplerate/1000);
        int end = (milliseconds <= 0) ? size :
(int) ((float) (startmillisecond + milliseconds)*samplerate/1000);
        data = new float[size];
        int index = 0;
        /*Concatenate*/
        foreach (float[] chunk in chunks)
        {
            Array.Copy(chunk, 0, data, index, chunk.Length);
            index += chunk.Length;
        }
        /*Select specific part of the song*/
        if (start ! = 0 || end ! = size)
        {
            float[] temp = new float[end – start];
            Array.Copy(data, start, temp, 0, end – start);
            data = temp;
        }
        else
throw new Exception(Bass.BASS_ErrorGetCode().ToString());
        return data;
}
```

下面的源代码片段 (使用标准的低通滤波器) 将采样率从 44100Hz 降低至 5512Hz:

```
/// <summary>
/// Low pass filter for downsampling 44100Hz to 5512Hz
/// </summary>
readonly double [] _lpFilter44 =
{
      –6.4796e–04, –1.4440e–03, –2.7023e–03, –4.4407e–
03,–6.1915e–03,–6.9592e–03,
    –5.3707e–03, 2.3907e–18, 1.0207e–02, 2.5522e–02,
 4.5170e–02, 6.7289e–02,
      8.9180e–02, 1.0778e–01, 1.2027e–01, 1.2467e–01,
1.2027e–01, 1.0778e–01,
      8.9180e–02, 6.7289e–02, 4.5170e–02, 2.5522e–02,
1.0207e–02, 2.3907e–18 ,
      –5.3707e–03, –6.9592e–03, –6.1915e–03, –4.4407e–
03, –2.7023e–03, –1.4440e–03,
      –6.4796e–04
};
/// <summary>
/// Convolve a sample with a filter at a given point
/// </summary>
public static float ConvolvePoint (float [] samples, int
startIndex, double[] filter)
{
    float sum = 0;
    int len = filter.Length;
    for (int i = 0; i < len; i++)
        sum += (float) (samples[i + startIndex]*filter[i]);
    return sum;
}
/// <summary>
/// Downsample a signal from 44100Hz to 5512Hz
/// </summary>
public float[] Downsample(float[] samples)
{
    const int multiplier = 8;
    int nsamples = samples.Length;
    int newnsamples = nsamples/multiplier – 3;
    float[] newsamples = new float[newnsamples];
    for (int i = 0; i < newnsamples; i++)
        newsamples[i] = ConvolvePoint(samples, multiplier*i,
_lpFilter44);
    return newsamples;
}
```

下面的源代码片段完成声轨的转换:

```
/// <summary>
/// Mono conversion of a stereo signal decoded in bytes
/// </summary>
public byte[] StereoToMono(byte[] data)
{
    byte[] mono = new byte[data.Length / 2];
    for (int i = 0; i < mono.Length / 2; i++)
    {
        int left = (data[i * 4] << 8) | (data[i * 4 + 1] & 0xff);
 int right = (data[i * 4 + 2] << 8) | (data[i * 4 + 3] & 0xff);
        int avg = (left + right) / 2;
        short m = (short)avg; /*Mono is an average between 2
channels (stereo)*/
        mono[i * 2] = (byte)((short)(m >> 8));
        mono[i * 2 + 1] = (byte)(m & 0xff);
    }
    return mono;
}
```

3.2 创建频谱

在将信号预处理成 5512Hz 的 PCM 信号后，算法的第二个步骤就是构建音频输入的频谱。在数字信号处理中，频谱是一个信号随着时间变化的谱线密度的图像。将信号转换成频谱包括几个子步骤：首先，为了获取信号的谱线密度的时域变化，需要将信号切分成重叠的帧，然后将每帧进行一次快速傅里叶变换 (Fast Fourier Transform) 处理。根据其他音频指纹研

究成果的数据，处理过程使用的参数有：音频帧的时长是371ms (2048个样本)，变换间隔11.6ms (64个样本)。

这种length/spacing联合参数的一个最重要的结果是：频谱随着时间的变化变缓，为不确定时间位置提供鲁棒性。需要指出的是，在进行这种变换之前，输入信号的每个切片汉恩窗口 (Hann Window) 函数赋予一定权重值，进而将映射 F 的值域压缩到一个指定的范围 (通常是 [-1,1])。

下面的源代码片段展示了这种窗口函数：

```
// <summary>
      /// Gets the corresponding window data (according to
///  specified length)
/// </summary>
public double [] GetWindow(int length)
{
   double[] array = new double[length];
   //Hanning window
   for (int i = 0; i < length; i++)
 array[i] = 0.5 * (1 - Math.Cos(2 * Math.PI * i / (length - 1)));
     return array;
}
```

在完成每个切片的傅里叶后，输出频谱经过剪辑，318Hz~2000Hz部分被保留下来。根据研究，该频带足以表征一个音频文件的可感知内容。该频域也是公认的人类听觉系统的标准频带空间。然而，人类个体能听到的频度范围又很大地与环境因素相关，因此一般的可接受的听觉频率范围为20Hz~20000Hz。

回到算法中来，为了最小化输出维数 (每帧在经过FFT变换后是一个尺寸为 [0-1025] 的向量)，特定范围的318Hz~2000Hz的频带经过编码，2048个时域样本被压缩成1025个频域样本，它们是32个栅格的对数刻度的和。因此，算法中每帧的压缩因子为2048/32=64。总的说来，为了简化计算才选择使用对数空间，因为频带边缘位置对如此粗粒度的采样产生强烈影响。

下面的源代码片段用于产生频谱：

```
/// <summary>
/// Create log-spectrogram
/// </summary>
public float [][] CreateLogSpectrogram (IAudio proxy, string
filename,
            int milliseconds, int startmilliseconds)
{
   //read 5512 Hz, Mono, PCM, with a specific proxy
    float [] samples = proxy.ReadMonoFromFile (filename,
SampleRate,
                milliseconds, startmilliseconds);
   NormalizeInPlace(samples);
   int overlap = Overlap;
   int wdftSize = WdftSize;
   int width =  (samples.Length - wdftSize)/overlap; /*width
of the image*/
   float[][] frames = new float[width][];
   float[] complexSignal = new float[2*wdftSize];  /*even -
Re, odd - Img*/
   for (int i = 0; i < width; i++)
   {
//take 371 ms each 11.6 ms (2048 samples each 64 samples)
      for (int j = 0; j < wdftSize /*2048*/; j++)
      {
complexSignal[2*j] = (float) (_windowArray[j]*samples[i*overlap
+ j]);
         /*Weight by Hann Window*/

         complexSignal[2*j + 1] = 0;
      }
      //FFT transform for gathering the spectrum
Fourier.FFT(complexSignal, wdftSize, FourierDirection.Forward);
      frames[i] = ExtractLogBins(complexSignal);
   }
    return frames;
}
```

3.3 小波分解

一旦获得对数频谱 (在318Hz~2000Hz范围内分布着32个栅格，每个11.6秒)，第四步是将整个频谱图分割成切片(128*32的子谱图)。从编程层面来说，在构造了对数频谱之后，将整个图像数据编码成一个 [n][32]二维浮点型 (float) 数组，其中n等于信号长度 (ms) 除以11.6ms的值，32表示频带的数量。在得到子谱图之后，下一步将对它们进行哈尔小波分解 (Haar Wavelet Decomposition) 操作。小波是一种有限振幅、快速衰减的振荡波形。经过精心制作的小波信号具备独特的性质，可用于信号处理。它们可以与卷积技术结合使用，作为某个信号的一部分用于从该信号提取信息。

回到算法中来，对于每个频谱图，为其计算一个小波签名。值得一提的是，根据各种研究成果，无需跟踪小波的幅度，只需要关注其正负性 (+/-)。因此该架构使用如下签名编码：负数01，正数10，零00。这种比特向量编码具备两个重要特征：稀松性和抗退化性。

下面的源代码片段实现哈尔小波分解：

```
/// <summary>
///   Haar wavelet decomposition algorithm
/// </summary>
public class HaarWavelet : IWaveletDecomposition
{
   #region IWaveletDecomposition Members
   /// <summary>
   ///   Apply Haar Wavelet decomposition on the frames
   /// </summary>
   /// <param name = "frames">Frames to be decomposed</
/// param>
   public void DecomposeImageInPlace(float[][] frames)
```

```
    {
        DecomposeImage(frames);
    }
    #endregion
    /// <summary>
      ///  Decomposition taken from Wavelets for Computer
/// Graphics
    /// </summary>
    private static void DecomposeArray(float[] array)
    {
        int h = array.Length;
        for (int i = 0; i < h; i++)
/*doesn't introduce any change in the final fingerprint image*/
            array[i] /= (float)Math.Sqrt(h);
            /*because works as a linear normalizer*/
        float[] temp = new float[h];
        while (h > 1)
        {
            h /= 2;
            for (int i = 0; i < h; i++)
            {
temp[i] = (float)((array[2 * i] + array[2 * i + 1]) / Math.Sqrt(2));
temp[h + i] = (float)((array[2 * i] – array[2 * i + 1]) / Math.Sqrt(2));
            }
            for (int i = 0; i < 2*h; i++)
            {
                array[i] = temp[i];
            }
        }
    }
    /// <summary>
                ///  The standard 2–dimensional Haar wavelet
/// decomposition
    ///  involves one–dimensional decomposition of each row
        ///  followed by a one–dimensional decomposition of
/// each column of the result.
    /// </summary>
    private static void DecomposeImage(float[][] image)
    {
        int rows = image.GetLength(0); /*128*/
        int cols = image[0].Length; /*32*/
         for  (int row = 0; row < rows /*128*/; row ++ ) /
*Decomposition of each row*/
            DecomposeArray(image[row]);
        for  (int col = 0; col < cols /*32*/; col ++ ) /
*Decomposition of each column*/
        {
            float[] column = new float[rows];
  /*Length of each column is equal to number of rows*/
   for (int row = 0; row < rows; row++)
  column[row] = image[row][col]; /*Copying Column vector*/
            DecomposeArray(column);
            for (int row = 0; row < rows; row++)
                image[row][col] = column[row];
        }
      }
}
```

到目前为止，都是创建音频指纹所需的准备事项。下面才是真正开始创建工作：首先，创建一个对数频谱；然后，将频谱图像切分成 N*32 的子谱图，并返回音频指纹列表。值得强调的是，两枚连续指纹的距离（步幅）可通过 IStride 接口实现。在算法中，关于步幅的选取不同：指纹库创建时采用静态的 928ms 步幅来产生两枚连续的指纹；在指纹查询中使用 0–46ms 的随机距离。

下面的源代码片段创建一个真实的音频指纹列表：

```
/// <summary>
/// Create fingerprints according to the Google's researchers
/// algorithm
/// </summary>
public List<bool []> CreateFingerprints (IAudio proxy, string
filename,
    IStride stride, int milliseconds, int startmilliseconds)
{
   float[][] spectrum =
    Cre ateLogSpectrogram (proxy, filename, milliseconds,
startmilliseconds);
   int fingerprintLength = FingerprintLength;
   int overlap = Overlap;
   int logbins = LogBins;
   int start = stride.GetFirstStride() / overlap;
   List<bool[]> fingerprints = new List<bool[]>();
   int width = spectrum.GetLength(0);
   while (start + fingerprintLength < width)
   {
      float[][] frames = new float[fingerprintLength][];
      for (int i = 0; i < fingerprintLength; i++)
      {
         frames[i] = new float[logbins];
         Array.Copy(spectrum[start + i], frames[i], logbins);
      }
      start += fingerprintLength + stride.GetStride() / overlap;
          WaveletDecomposition.DecomposeImageInPlace
(frames); /*Compute wavelets*/
       bool[] image = ExtractTopWavelets(frames);
       fingerprints.Add(image);
   }
    return fingerprints;
}
```

3.4 指纹的哈希处理

哈希函数在计算机科学中得到广泛应用的一个重要原因是：对于两个大型对象 X 和 Y 的比较，可以通过比较它们对

应的哈希值 $H(X)$和 $H(Y)$来完成。从概率的角度来说，这种比较的一致性高度吻合，即使是在冲突的情况也是如此。

一个音频指纹可以看成相应音频对象的缩微摘要。从数学的角度来说，一个指纹标记函数 F将包含大量比特的音频对象 X映射成一个有限压缩比特的音频指纹。因此，整个指纹提取机制 F可以看作一个哈希函数 H。

到目前为止，笔者的算法可以得到一个 8192 字节长的指纹列表。为了进一步压缩它们的长度，需更紧凑地表征每枚指纹。因此，需要开发一种方法来计算这些宽松比特向量的子指纹，称之为 Min-Hash 技术。该散列技术使用了一种称为 Jaccard 相似度的索引：它是一个取值 0 或 1 的数。当两个集合不相交时，取值 0；当它们相交时，取值为 1。Min-Hash 在该算法中的成功运用是基于该散列技术的一个重要特性：假设 C_1和 C_2代表两枚指纹，则 MinHash（C_1）与 MinHash（C_2）相等的概率与 Jaccard 相似度相等。

下面的源代码片段实现音频指纹的散列处理过程：

```
/// <summary>
/// Compute Min-Hash signature of a given fingerprint
/// </summary>
public int[] ComputeMinHashSignature(bool[] fingerprint)
{
    bool[] signature = fingerprint;
    int[] minHash = new int[_permutations.Count]; /*100*/
    for (int i = 0; i < _permutations.Count /*100*/; i++)
    {
        minHash[i] = 255;
        /*The probability of occurrence of 1 after
          position 255 is very insignificant*/
        int len = _permutations[i].Length;
        for (int j = 0; j < len /*256*/; j++)
        {
            if (signature[_permutations[i][j]])
            {
    minHash[i] = j; /*Looking for first occurrence of '1'*/
                break;
            }
        }
    }
return minHash; /*Array of 100 elements with bit turned ON if
        permutation captured successfully a TRUE bit*/
}
```

3.5 解决查找效率问题

前述段落中，采用傅里叶变换、哈尔小波分解以及 Min-Hash 散列机制对初始音频指纹向量进行一系列降维处理：从最初的子谱图 (128*32floats) 中收集到一个包含 100 个 8 比特整型数的向量来表示指纹本身。然而到目前为止，该架构缺乏并且急切需要一个在一个长度为 100 的向量空间中进行高效搜索的方法。

LSH (Locality Sensitive Hashing，局部敏感散列) 技术是一个非常重要的分类机制，用于查找最近邻居。事实证明，它执行必要的比较的次数合理适中并且具备噪声鲁棒性特质。KNN (k-nearest neighbors，最近邻居查找) 问题定义如下：给定一个有 K 个元素的数据集合，构建一个数据结构，对于任意查询点，报告离查询点最近的元素。KNN 问题应用在多个领域，比如数据压缩、数据库与数据挖掘、信息检索、音频与视频库、机器学习、模式识别、数据统计与分析等。

通常，对每个对象感兴趣的特征集可以表示为一个节点向量 R^d，并使用距离度量来测算对象的相似性。本质上，这种测算就是为查询对象执行索引或相似性查找的基本问题。对象特征的数量 (向量的维度) 是任意的。例如，开发者可以使用一个 100000 维空间中的一个向量来表示一个 1000×1000 的图像，每个像素对应一个维度。

一方面，为了顺利地引入 LSH 技术，需要使用几个哈希函数将向量中的元素点进行散列处理。通过散列查询点以及检索包含该查询点的存储元素来确定离查询点最近的邻居。事实上，最近邻居问题是一个优化问题，其目标是：对于某个目标函数 (在本架构中是距离度量)，寻找一个极小值点。为了简化标记，定义：对于点 p 是点 q，如果 p 和 q 之间的最远距离为 R，则称 p 是 q 的一个 R 接近邻居。

另一方面，KNN 算法会不仅返回精确的匹配，也会返回与查询参数类似的对象。比如，假设需要从一个电子书库中查找包含“port”词汇的所有书籍，则 KNN 搜索算法不仅返回精确匹配“port”的实例，也会返回“support”、“report”、“airport”、“portfolio”等类似词汇的实例。这种模糊匹配方法在解决某些特定问题时非常有用。在本架构中，它不仅可以查找相似音频对象，也提高了搜索速度。

下面的源代码片段构建了基于 Min-Hash 指纹值的 LSH 表结构：

```
/// <summary>
///  Compute LSH hash buckets which will be inserted into
/// hash tables.
///  Each fingerprint will have a candidate in each of the hash
/// tables.
/// </summary>
public Dictionary<int, long> GroupMinHashToLSHBuckets(int
[] minHashes,
  int numberOfHashTables, int numberOfMinHashesPerKey)
{
    Dictionary<int, long> result = new Dictionary<int, long>();
     const int maxNumber = 8; /*Int64 biggest value for
MinHash*/
    if (numberOfMinHashesPerKey > maxNumber)
                                throw new ArgumentException ("
numberOfMinHashesPerKey cannot be bigger than 8");
```

```
for (int i = 0; i < numberOfHashTables /*hash functions*/; i++)
   {
       byte[] array = new byte[maxNumber];
       for   (int j = 0; j < numberOfMinHashesPerKey /*r min
hash signatures*/; j++)
       {
array[j] = (byte)minHashes[i * numberOfMinHashesPerKey + j];
       }
       long hashbucket = BitConverter.ToInt64(array, 0);
//actual value of the signature
        result.Add(i, hashbucket);
   }
    return result;
}
```

3.6 检索

在 Min-Hash 指纹被分离到 LSH 表之后，将完成算法架构的最后一个步骤：从音频文件中提取指纹签名。整个检索进程的前部分与指纹创建过程类似。尽管指纹签名可以选择全部音频，但是为了在指纹精度和处理时间之间取得平衡，可以将音频文件的某些时帧作为指纹构造的依据。该过程一直循环，直到指纹哈希值分离到 LSH 表中。一旦完成，就可以将查询请求发送到指纹库引擎，引擎会查找有多少散列表包含指定的哈希值。对于阈值 v，如果获取的结果多于 v，表明有多于 v 的哈希表返回了相同的指纹，则该指纹可视为一个潜在匹配。为了得到最佳候选，需要对所有的潜在匹配进行分析。

下面的源代码片段实现了这种检索模式：

```
/// <summary>
/// Get tracks that correspond to a specific hash signature
///  and pass the threshold value
/// </summary>
public Dictionary<Track, int> GetTracks(int [] hashSignature,
int hashTableThreshold)
{
Dictionary<Track, int> result = new Dictionary<Track, int>();
   for (int i = 0; i < _numberOfHashTables; i++)
    {
       HashSet<Track> voted = new HashSet<Track>();
       for (int j = 0; j < _numberOfHashTables; j++)
       {
          if (_hashTables[i].ContainsKey(hashSignature[j]))
          {
             var tracks = _hashTables[i][hashSignature[j]];
             foreach (var track in tracks)
             {
                if (voted.Contains(track)) continue;
                if (! result.ContainsKey(track))
                   result[track] = 1;
                else
                   result[track]++;
                voted.Add(track);
             }
          }
       }
   }
   Dictionary<Track, int> filteredResult =
                        result.Where  (item  =>  item.Value  >=
hashTableThreshold).ToDictionary(
      item => item.Key, item => item.Value);
      return filteredResult;
}
```

4 结语

简要地阐释了音频指纹的一些核心概念，提出了一个音频指纹构建与搜索的算法架构，使用 C# 编程很好地实现了这一架构，为进一步的具体应用开发打下坚实的基础。

（收稿日期：2013-01-30）

（上接第 17 页）

执行效果如图 5 所示。

图 5 运行效果

3 结语

ERP 系统软件 SAP 的成功上线不是企业信息化建设的终点，而是新里程的起点，建设信息化难，促进信息系统深入应用更难，如何持续地推进信息系统建设，让 ERP 系统为企业发展发挥作用，让每名员工从信息化建设中得到切实的好处是我们必须不断思考的问题，开发 SAP 外围接口应用程序，是国际先进管理理念与企业业务实际相结合一种方案。

参考文献

[1] 戈贝尔. SAP 企业门户技术和编程. 东方出版社，2005.

[2] George W.Anderson. SAP 实施大全. 段大为，王丹，译. 人民邮电出版社，2012.

[3] 唐骏华. SAP ABAP 实用程序开发. 机械工业出版社，2010.

（收稿日期：2012-10-12）

哈希结构模拟文件系统

杨 东

摘 要：使用哈希结构模拟了文件系统中的文件记录，讲述了以哈希算法实现文件记录的保存、查找和删除的方法，以算法流程图与C语言结合的方式演示了模拟的过程。

关键词：哈希结构；哈希算法；文件系统；文件记录

1 哈希结构

哈希结构（Hash Structure）又称散列结构，适用于定长记录按键值随机查找的访问方式。其思想是通过某个哈希函数来确定一个记录在存储设备上的存储位置。对于有些文件，如果存在一个域，其值对于所有记录均不相同，便可以用来作为检索键（key）。定义一个哈希函数 Hash ()，用此函数作用于检索键 key，便可得到一个地址 addr=Hash (key)，将此地址作为该记录在存储设备上的存储位置。哈希结构如图 1 所示。

图 1 哈希结构的记录形式

若存在两个键 key1 和 key2，使得 addr1=Hash (key1)，addr2=Hash (key2)，则需要对此冲突进行处理，处理冲突最常用的方法是线性探查法。

线性探查法的原理为：当冲突发生时，由冲突地址处开始向后顺序探查（需探查整个空间），找到第一个空闲位置，并将记录保存于该处。这样，addr=Hash (key) 便成了该记录检索的起始位置。为了找到记录位置，还需要记录冲突次数，因而需要在记录中增加两个域：一个是冲突计数，用于标识冲突次数；另外一个是空闲标志，用于标识本记录是否空闲。

2 数据定义

2.1 数据结构

定义哈希结构（Hash Structure）模拟文件记录，其中包含检索键、冲突次数、空闲标志 3 个域。代码如下：

```
#define MAXSIZE 8
#define ARRAY 5
typedef struct HashTable{//哈希结构
    int count;//冲突次数
    int empty;//空闲标志,1-空闲,0-占用
    int key;
};
 struct HashTable HT[MAXSIZE];//哈希表,模拟存放 8 个文件
//记录
```

2.2 哈希函数

哈希函数的实现方式有多种，如直接定址法、数字分析法、除留取余法等，在此选择易于实现的除留取余法。取关键字被某个不大于哈希表表长 m 的数 p 除后所得余数为哈希地址，即：

Hash（key）=key MOD p，p≤m

代码实现如下：

```
int Hash(int key)//哈希函数
{
    return key%MAXSIZE-1;
}
```

2.3 系统初始化

对预先定义的哈希表进行初始化。预先接收 5 个输入的检索键，置冲突计数为 1，空闲字段为 0，并在表中预留 3 个空闲空间，供后续测试使用。代码实现如下：

```
void Init()//初始化函数
{
int i;
printf("请输入 5 个整数:");
for(i=0;i<MAXSIZE;i++)
{
    if(i<ARRAY)
    {
        HT[i].count=1;
        HT[i].empty=0;
        scanf("%d",&HT[i].key);
    }
```

```
        else
        {
            HT[i].count=0;
            HT[i].empty=1;
        }
    }
}
```

3 保存记录

文件系统中要存入新记录时，使用哈希函数对记录的检索键 key 进行运算，获得应存入的哈希表空间的地址，然后在哈希表中检测该地址空间是否已经存入内容（空闲标志是否为 0），若该空间已经被占用，依次选取下一条地址检测是否冲突，直至找到一个空闲的空间，将该记录保存至表中。保存记录的算法流程图如图 2 所示。

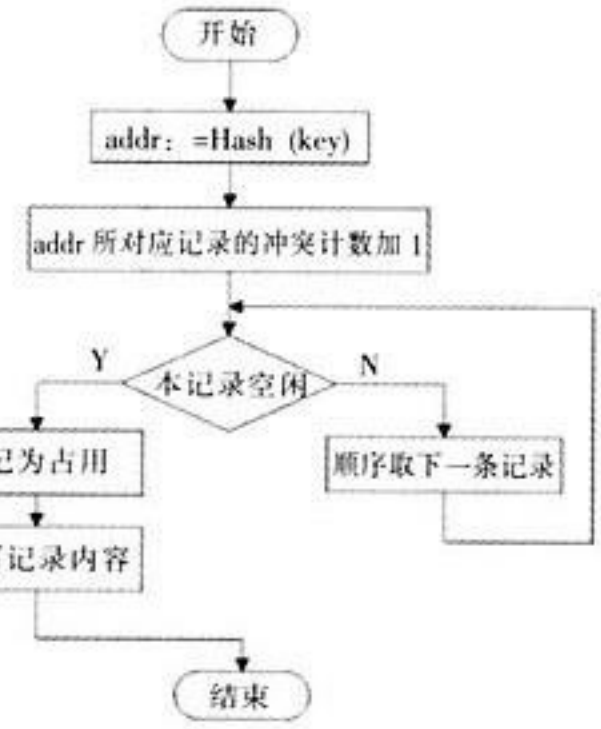


图 2 保存记录的算法流程图

代码实现如下：

```
int InsertHash(int key)
{//保存记录
    int num=1;//判断次数
    int addr=Hash(key);//计算哈希值
    HT[addr].count++;//addr 所对应的记录数加 1
    while(HT[addr].empty==0)//本记录不空
    {
        addr++;
        if(addr==MAXSIZE)
        {//至表尾,从头开始
            addr=0;
        }
        if(num==MAXSIZE)
        {//整个表比较一遍还未找到
            break;
        }
        num++;//比较次数加 1
    }
    if(HT[addr].empty==1)
    {
        HT[addr].empty=0;
        HT[addr].key=key;
        return 1;//存储记录
    }
    else
    {
        return 0;//Hash 表已满
    }
}
```

4 查找记录

查找某个文件记录的存放位置时，首先利用哈希函数取得地址，对该地址空间的记录进行判定，查看空间是否空闲，记录的检索键是否与当前查找的记录相同等，实现过程如图 3 所示。

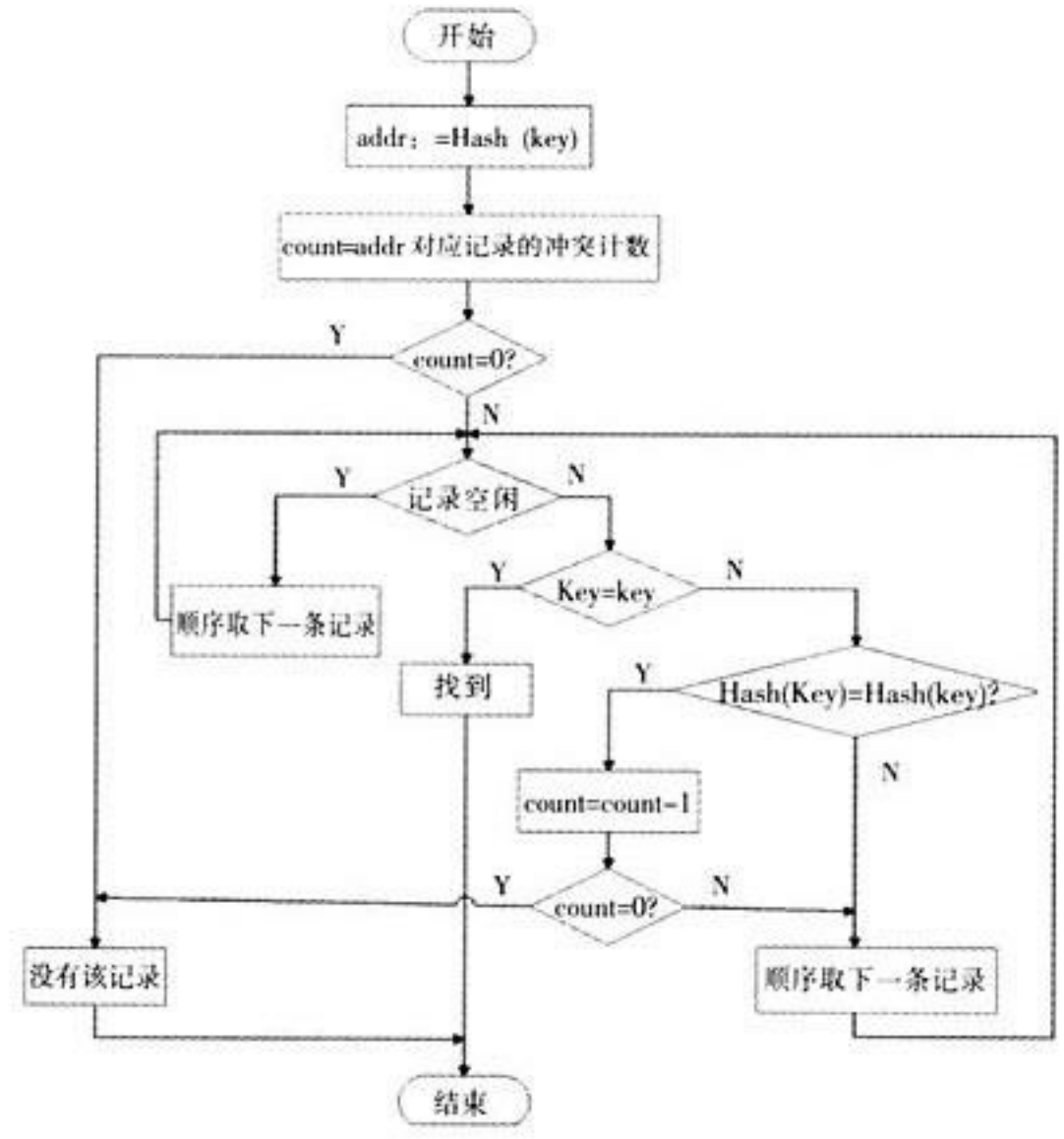


图 3 查找记录的算法流程图

代码实现如下：

```
int SearchHash(int key)
{//找到返回记录所在位置,未找到返回-1
    int num=1;//判断次数
    int addr=Hash(key);
    int count=HT[addr].count;
    if(count==0)
    {//无记录
        return -1;
    }
```

```
else
{
    while(1)
    {
        if(HT[addr].empty==1)
        {//记录空闲
            addr++;
            if(addr==MAXSIZE)
            {//至表尾,从头开始
                addr=0;
            }
            if(num==MAXSIZE)
            {//整个表比较一遍还未找到
                return -1;
            }
            num++;//比较次数加 1
    }
    else if(HT[addr].empty==0)
    {//记录非空闲
        if(HT[addr].key==key)
        {
            return addr;//找到位置
        }
        else
        {
            if(Hash(HT[addr].key)==Hash(key))
            {
                count--;
                if(count==0)
                {
                    return -1;//无记录
                }
                else
                {
                    //取下一条记录
                    addr++;
                    if(addr==MAXSIZE)
                    {//至表尾,从头开始
                        addr=0;
                    }
                    if(num==MAXSIZE)
                {//整个表比较一遍还未找到
                        return -1;
                    }
                    num++;//比较次数加 1
                }
            }
            else
            {
                //取下一条记录
                addr++;
                if(addr==MAXSIZE)
                {//至表尾,从头开始
                    addr=0;
                }
                if(num==MAXSIZE)
                {//整个表比较一遍还未找到
                    return -1;
                }
                num++;//比较次数加 1
            }
        }
    }
}
```

5 删除记录

若要删除某个记录，需先行查找该记录所在的位置，若记录存在，仅需将该记录对应的空间空闲标志为置 1，冲突计数减 1 即可，原空间所存放的内容无需清除，当下次需要在该空间再存入新内容的时候，将原先的内容覆盖即可。删除记录的流程如图 4 所示。

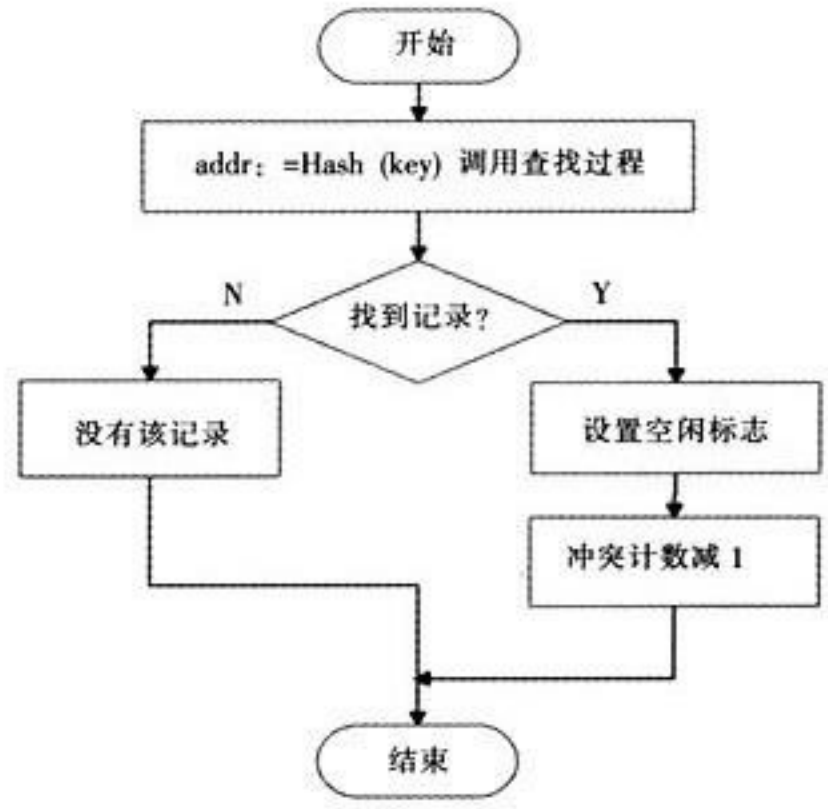


图 4 删除记录的算法流程图

代码实现如下：

```
int DelHash(int key)
{
    int addr=Hash(key);
    int pos=SearchHash(key);
    if(pos! =-1)
    {//找到记录
        HT[pos].empty=1;
        HT[pos].count--;
        return 1;
    }
```

（下转第 39 页）

基于 C# 的书法字典设计与实现

张中红

摘　要：基于 C# 编程，设计并实现了一个对书法字体图片检索与显示的书法字典程序。书法字库以图片形式保存，以文件夹形式存储，具有易于扩充，便于检索的特点。同时提供了字库图片的提取、裁剪、改变颜色、改变大小的处理方法，便于使用者方便地对字库进行扩充。

关键词：C# 语言；书法；图片；裁剪；缩放

1　引言

书法是中国特有的汉字写法的一种传统艺术。

汉字，是汉文化圈广泛使用的一种文字，也是上古时期各大文字体系中唯一传承至今，且连续使用时间最长的主要的文字。中国历代皆以汉字为主要官方文字。

中国汉字是古代劳动人民在生产生活中，观察世间万物创造出来的。中国汉字是一种象形符号，具备音形义的特点，经过几千年的发展，演变成了现在的文字。中国古代人发明了毛笔，用来进行文字书写，于是便产生了书法。

从狭义讲，书法是指用毛笔书写汉字的方法和规律。包括执笔、运笔、点画、结构、布局（分布、行次、章法）等内容。从广义讲，书法是指语言符号的书写法则。换句话说，书法就是指按照文字特点及其涵义，以其书体笔法、结构和章法写字，使之成为富有美感的艺术作品。

汉字书法为汉族独创的表现艺术，被誉为无言的诗，无行的舞，无图的画，无声的乐。书法是中华文化艺术宝库中具有独特魅力的瑰宝，它有着悠久的历史和优良的传承，其绚烂夺目的光芒更让人叹为观止。鲁迅先生曾说过：“视文字为美观是华夏之独特”。

如今，随着社会的发展，人民生活水平的提高，喜爱书法这门独特的艺术的人越来越多，不但是一些中老年人喜爱书写艺术，也有越来越多的青少年加入学习书法艺术的行列。然而学习书法，是一个勤学多练、循序渐进的过程，初学者要先经过一个临帖学习的过程。

初学者临摹字帖，首要的一点是要临得像，要掌握书法家的用笔方法和字体结构。中国历代有很多其名的书法家，留下了无数的书法瑰宝，是学习书法的临摹范本。现在市面上也有很多的书法字帖，供初学者进行临习书写，但是使用这些字帖却有些不便：一是大部分的字帖，一般一本字帖只是作者其中的一部作品，如颜真卿的《颜勤礼碑》，王羲之的《兰亭序》等，其中的文字数量有限，临完字贴也只能掌握字帖中提供的文字，字帖外的字还是不知道怎么写；二是在字帖中要找到一个字，要翻遍字帖，查找不便；三是使用字贴，一般只读到一个书法家的一部作品，无法学习更多书法家的字，无法博采众长，融会贯通；四是字帖质量不一，有的是拓本翻印，字迹模糊不清，也没有米字格，对字体定位不准，有的是其他书家摹本，已不是原书法家的真迹，失去了原书法家作品的神韵；五是字帖价值高昂，对初学者来说，买几本字帖还可以承受，要每本都买就不现实了。

因此，编制一书法字典程序，搜集历代书法名字作品真迹，对其字迹进行处理、放大显示，将会对书法爱好者练习书法有较大的帮助作用。

2　书法字典功能

2.1　书法字典管理

本书法字典不是一个商业化的软件作品，而是一个对书法爱好者起到帮助作用的工具，因此，本书法字典只是一个对书法字体图片进行管理和检索的管理程序，不提供书法字库，所有字库需要使用者根据需要自行搜集。

2.2　主要功能

本书法字典的主要功能分为两部分，如图 1 所示。

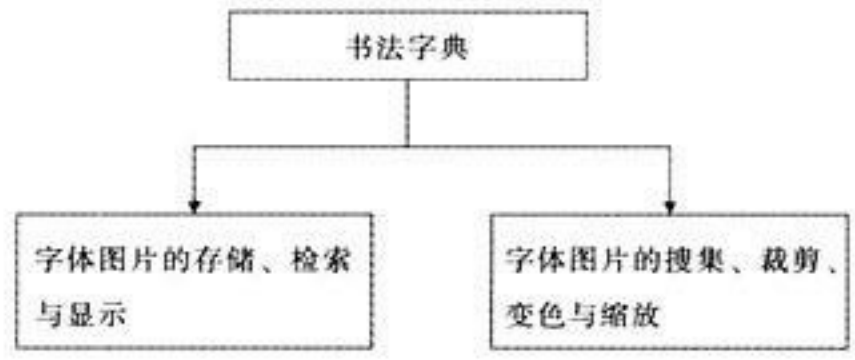


图 1　书法字典的整体架构

2.3　图片存储

本书法字典对字体图片的存储使用文件夹形式。检索字体时，书法字典程序对指定的文件夹进行检索，即可找到对应的字体图片。对字体图片进行处理，与米字格合成，并进行显

示。

2.4 图片来源

书法字典所需要的字体图片，既可以从字帖中扫描得来，也可以从网上下载得来，对字体图片文件处理后，按指定的文件名格式存入相应文件夹，即可方便地实现对字体库的扩充。

2.5 图片处理

字体图片处理功能，既可对单个图片进行处理，也可对多个图片进行批量处理，还可以对指定网址的图片进行批量下载并处理。

3 书法字典功能实现

3.1 书法字典界面

书法字典的界面如图 2 所示。在字体输入框内输入要找的文字，也可以指定书法家，还可以指定字体，进行复合检索。检索到要找的字体后，在预览窗口内以缩略图形式显示所找到的所有字体文件，并在字体显示窗口中显示找到的第一个字体文件。

图 2 书法字典的用户界面

在预览窗口内找到想观看的字体后，点击该字体文件，即可在显示窗口内以大字体显示指定的字体。

3.2 书法字典检索

字体检索时，根据指定的字体类型和作者姓名，在字体图片文件夹内进行查找。如果没有找到指定的字体图片，则提示没有找到，如果找到该字体图片，则将字体图片文件以缩略图形式显示在预览窗口内，并在显示窗口内显示查找到的第一个字体文件。

3.2.1 字体类型获取函数

根据单选按钮，获取指定的字体类型。

代码如下：

```
private string GetSpecFont()
{
    string returnvalue = "";
    if (radioButtonAll.Checked) returnvalue = "全";
    else if (radioButtonX.Checked) returnvalue = "行";
    else if (radioButtonK.Checked) returnvalue = "楷";
    else if (radioButtonL.Checked) returnvalue = "隶";
  else if (radioButtonZ.Checked) returnvalue = "篆";
 else if (radioButtonC.Checked) returnvalue = "草";
    return returnvalue;
}
```

3.2.2 字体图片查找函数

根据指定的字体类型和作者，在字体图片文件夹下的所有子目录中递归查找指定的字体图片文件，dir 参数为指定的目录：

```
    public void FindFile(string dir)
    {
        //在指定目录及子目录下查找文件,在 listBox1 中列出
//子目录及文件
        DirectoryInfo Dir = new DirectoryInfo(dir);
        try
        {
            if (currfontpath == fontpath)//如果是全部字体,则
//读取指定目录下的字体文件
            {
                foreach (DirectoryInfo d in Dir.GetDirectories())
//查找子目录
                {
                    FindFile(dir + "\\" + d.ToString());
                }
            }
            else if (! Directory.Exists(currfontpath))//如果指定
//文件不存在,则提示
            {
                label1.Text = "提示:当前文字还未收入！";
            }
            else//如果不是全部字体,而是单一字体,则读取指
//定目录下的字体文件
            {
                foreach (FileInfo f in Dir.GetFiles("*.*")) //查找
//当前目录下的全部文件
                {
                    if (f.Name.Contains('-'))//如果文件名中含
//有-,说明是字体文件
                    {
                        //以-分隔字体文件名
                string[] fontA = f.Name.Split(new char[] { '-' });
                        string specfont = GetSpecFont();
//如果不指定作者,关键字取数组中第一个元素
                        if (textBoxAuthor.Text == "")
                        {
                  if (specfont == "全" || specfont == fontA[0])
                            {
                //如果是查找到的第一个字,则显示到 picturebox 内。
                                if (firstinLV == "")
                                {
                    firstinLV = dir + "\\" + f.ToString();
```

```
                }
                //在 listBox1 中添加文件名
            listView1.Items.Add(f.ToString(), imgindex);
/* 图片列表为固定相等长宽,即正方形,为了在缩略图中表现字体的真正长宽比例,新建一个边长等于字体长边的正方形,将字体画入正中间,然后生成此正方形的缩略图,加入图片列表,即可在程序中表现字体的真正形状 */
    Image tmpimg = Image.FromFile(dir + "\\" + f.ToString());
                int longborder;
        if (tmpimg.Width > tmpimg.Height)
                    longborder = tmpimg.Width;
                else
                    longborder = tmpimg.Height;
  Bitmap tmpbmp = new Bitmap(longborder, longborder);
            //在背景图片上创建画板,用来写字体
            Graphics g = Graphics.FromImage(tmpbmp);
                //将字体的全部写入画板上
                    g.DrawImage (tmpimg, new
Rectangle ((longborder - tmpimg.Width) / 2, (longborder - tmpimg.Height) / 2, tmpimg.Width, tmpimg.Height), 0, 0, tmpimg.Width, tmpimg.Height, GraphicsUnit.Pixel);

                //生成缩略图的 imagelistlarge
            imageListLarge.Images.Add (tmpbmp.
GetThumbnailImage(65, 65, myCallback, IntPtr.Zero));

                    imgindex++;
                    findfontnum++;
                }
            }
        else //如果指定了文字和作者,则进行综合查找
            {
                if  ((specfont == "全" || specfont ==
fontA[0]) && textBoxAuthor.Text == fontA[2])
                {
                    //如果是查找到的第一个字,则显示
//到 picturebox 内。
                    if (firstinLV == "")
                    {
          firstinLV = dir + "\\" + f.ToString();
                    }
                     listView1.Items.Add  (f.ToString (),
imgindex);    //listBox1 中添加文件名
                //新建一个边长等于字体长边的正方形,将字体
//画入正中间,然后生成此正方形的缩略图,加入图片列表,即可
//在程序中表现字体的真正形状
  Image tmpimg = Image.FromFile(dir + "\\" + f.ToString());
                    int longborder;
            if (tmpimg.Width > tmpimg.Height)
                    longborder = tmpimg.Width;
                else
                    longborder = tmpimg.Height;
  Bitmap tmpbmp = new Bitmap(longborder, longborder);
            //在背景图片上创建画板,用来写字体
            Graphics g = Graphics.FromImage(tmpbmp);
                //将字体的全部写入画板上
                    g.DrawImage (tmpimg, new
Rectangle ((longborder - tmpimg.Width) / 2, (longborder - tmpimg.Height) / 2, tmpimg.Width, tmpimg.Height), 0, 0, tmpimg.Width, tmpimg.Height, GraphicsUnit.Pixel);
                //生成缩略图的 imagelistlarge
                imageListLarge.Images.Add (tmpbmp.
GetThumbnailImage(65, 65, myCallback, IntPtr.Zero));
                    imgindex++;
                    findfontnum++;
                }
            }
        }
        catch (Exception e)
        {
            MessageBox.Show(e.Message);
        }
    }
```

3.2.3 字体检索功能

指定检索条件后，点击查找按钮，执行检索与显示。代码如下：

```
  private void buttonFind_Click(object sender, EventArgs e)
    {
        //将预览窗口和缩略图列表清空,缩略图的 imagelist
//序号置 0。
        listView1.Clear();
        imageListLarge.Images.Clear();
        imgindex = 0;

        //查找到的字数置零。
        findfontnum = 0;
        //把第一个字置空,准备记录查找到的第一个字,以便
//在 picbox 中显示。
        firstinLV = "";
        //根据当前点选的字体,取得字体所在的路径
        currfontpath = fontpath + "\\" + textBoxFind.Text;
        //查找并显示字体
        FindFile(currfontpath);
        ViewPic(firstinLV);
        label1.Text = "提示:查找到" + findfontnum.ToString()
+ "个""+textBoxFind.Text+""字。";
    }
```

3.3 字体图片的显示

显示窗口定义初始窗口为 600×600 大小。找到字体文件后，在内存中将字体图片进行按比例缩放，并计算后将文字写

入背景米字格的中心位置，进行合成。然后将合成的图像写入显示区：

```
private void ViewPic(string filepathname)
{
    if (filepathname ! = "")
    {
        string fontname = filepathname.Substring(filepathname.LastIndexOf('\\') + 1);
        if (fontname.Contains('-'))
        {
    string[] fontnameA = fontname.Split(new char[] { '-' });
            labelTS.Text = "当前文字:" + fontnameA[1] + "字体:" + fontnameA[0] + "书　作者:" + fontnameA[2];
        }
        //找到字体文件后,显示第一个字体。
        int fonttop, fontleft, fontwidth, fontheight;//字体显示时用的宽度和高度
        //在内在中创建临时画板,用以合成背景和字体
        Image tmpimg;//临时 img
        Image fontimg;//字体 img
        //读取背景图片
        //Bitmap bkbmp = new Bitmap("sfborder.png");
            tmpimg = Image.FromFile(Application.StartupPath + "\\sfborder.png");
        //根据计算得到的字体宽高,得到指定大小的字体缩略图片
        tmpimg = tmpimg.GetThumbnailImage(pictureBox1.Width, pictureBox1.Height, myCallback, IntPtr.Zero);
        //在背景图片上创建画板,用来写字体
        Graphics g = Graphics.FromImage(tmpimg);
        //读取指定的字体文件
        fontimg = Image.FromFile(filepathname);
        //对字体进行缩放,计算缩放后字体的宽和高。
        //如果是长形字体
        if (fontimg.Width < fontimg.Height)
        {
            fontheight = pictureBox1.Height - 100;
            fontwidth = (int)((fontimg.Width * fontheight / fontimg.Height));
        }
        else//如果是宽形字体
        {
            fontwidth = pictureBox1.Width - 100;
fontheight = (int)((fontimg.Height * fontwidth / fontimg.Width));
        }
        //根据计算得到的字体宽高,得到指定大小的字体缩略图片
        fontimg = fontimg.GetThumbnailImage(fontwidth, fontheight, myCallback, IntPtr.Zero);
        //计算字体写在画板上左上角的位置
        fonttop = (pictureBox1.Height - fontheight) / 2;
        fontleft = (pictureBox1.Width - fontwidth) / 2;
        //将字体的全部写入画板上
        g.DrawImage (fontimg, new Rectangle(fontleft, fonttop, fontimg.Width, fontimg.Height), 0, 0, fontimg.Width, fontimg.Height, GraphicsUnit.Pixel);
        //在 pictureBox 中间显示此字体图片
        pictureBox1.Image = tmpimg;
    }
}
```

3.4 字体图片的选择

点击预览窗口 listview 中的缩略图项目时，在显示区 pictureBox 中显示字体图片：

```
private void listView1_Click(object sender, EventArgs e)
{
    string [] fontA = listView1.FocusedItem.Text.Split(new char[] { '-' });
    currfontpathname = fontpath + "\\" + fontA [1] + "\\" + listView1.FocusedItem.Text;
    ViewPic(currfontpathname);
}
```

3.5 输入框与查找按钮状态控制

搜索框内容变动时，改变搜索按钮的模式：

```
private void textBoxFind_TextChanged (object sender, EventArgs e)
{
    if (textBoxFind.Text ! = "")
    {
        buttonFind.Enabled = true;
    }
    else
    {
        buttonFind.Enabled = false;
    }
}
```

光标在搜索框时，按回车键时，开始搜索：

```
private void textBoxFind_KeyPress (object sender, KeyPressEventArgs e)
{
    if (e.KeyChar == (char)Keys.Return)
    {
        buttonFind_Click(sender, e);
    }
}
```

光标在作者框时，按回车键时，开始搜索：

```
private void textBoxAuthor_KeyPress (object sender, KeyPressEventArgs e)
{
    if (e.KeyChar == (char)Keys.Return)
    {
```

```
            buttonFind_Click(sender, e);
        }
    }
```

3.6 窗口缩放控制

书法字典主窗口缩放时，将字体图片按比例相应缩放：

```
    private void Formmain_Resize (object sender, EventArgs e)
    {
        int tmplen;
        if (splitContainer1.Panel2.Width <= splitContainer1.Panel2.Height)
        {
            tmplen = splitContainer1.Panel2.Width - 30;
        }
        else
        {
            tmplen = splitContainer1.Panel2.Height - 30;
        }
        panelPic.Width = panelPic.Height = tmplen;
        pictureBox1.Width = tmplen-26;
        pictureBox1.Height = tmplen-26;
        ViewPic(currfontpathname);
    }
```

3.7 字体图片保存

点保存按钮后，将把字体文件和背景米字格的合成图保存成图片文件：

```
    private void buttonSaveAs_Click (object sender, EventArgs e)
    {
  if (saveFileDialog1.ShowDialog() == DialogResult.OK)
      {
   pictureBox1.Image.Save(saveFileDialog1.FileName);
      }
    }
```

3.8 字体图片打印

点打印按钮后，将把字体文件和背景米字格的合成图打印出来：

```
    private void printDocument1_PrintPage (object sender, System.Drawing.Printing.PrintPageEventArgs e)
    {
        e.Graphics.DrawImage(pictureBox1.Image, 20, 20);
    }
private void buttonPrint_Click(object sender, EventArgs e)
    {
        if (printDialog1.ShowDialog() == DialogResult.OK)
        {
            try
            {
                printDocument1.Print();
            }
            catch
            {  //停止打印
                printDocument1.PrintController.OnEndPrint(printDocument1, new System.Drawing.Printing.PrintEventArgs());
            }
        }
    }
```

4 字体图片文件的搜集与制备

4.1 字体图片文件处理的界面

字体图片文件的搜集与制备是书法字典的辅助工具，本工具可以辅助搜索字体图片文件，以及对字体图片文件进行裁剪、变色、缩放等处理。

字体图片文件处理界面如图 3 所示。

图 3 字体图片处理

4.2 字体图片的搜集

字体图片的搜集主要有两种方式，一是从字帖中，通过扫描或照相，得到字体图片，这种方法需要一些专业技能，并且效率较低，优点是自给自足，比较灵活，质量有保证。二是从网上下载字体图片。这种方法是使用已有的图片，信息量大，缺点是图片格式五花八门，图片尺寸大小不一，质量不能保证，并且有些图片涉及版权。

对于第一种方法视个人情况而定，不再赘述。此处简单介绍第二种方法。

在网络上有很多书法爱好者上传的历代名家的书法图片，供书法爱好者欣赏学习，也有一些网站，提供一些书法名家的书法墨宝图片。使用这些图片，可以省却搜集字帖，照相扫描的工作。当然，对于一些公司的有版权的图片，请谨慎使用。

对于单个或少数的字体图片，可以对字体图片逐一下载，然后进行处理。但对一些集合的多个字体图片，可以进行批量下载，减少繁琐重复的操作过程。

下面举一个从网络上批量下载字体图片的例子。

某网站的一个页面上有多个字体图片，逐一下载比较烦琐，对页面文件分析后，从里面解析出图片地址，并下载到本地，按规定的规则对图片文件进行重命名后保存到指定的地址。

批量下载字体图片文件代码如下：

```
private void button1_Click(object sender, EventArgs e)
{
    int fontnum = 0;//抓取的文字个数
    labelTS.Text = "";
    if (richTextBox1.Text ! = "")
    {
        string [] fonttype = richTextBox1.Text.Split(new
string[] { "书 ';var b=new Array(\"" }, StringSplitOptions.None);
        string[] fontword = fonttype[1].Split(new string[] {
"\");\nvar b0=[[" }, StringSplitOptions.None);
//给出的字体文件网络地址的字串,按规则拆分为数组,每组含
//两部分,第一部分是字体作者,第二部分是存放位置
        string[] srcStrA = fontword[1].Split(new string[] { "
'],[" }, StringSplitOptions.None);
        //以数组长度,即要处理字体的个数对进度条赋值。
        progressBar1.Minimum = 0;
        progressBar1.Maximum = srcStrA.Length;
        progressBar1.Value = 0;
        //遍历数组,抓取每个字体图片,并按命名规则保存
//到本地指定文件夹
        for (int i = 0; i < srcStrA.Length; i++)
        {
            string[] fontA = srcStrA[i].Split(new string[] { "',
" }, StringSplitOptions.None);
      string imgUrl = "http://sf.kdd.cc/" + fontA[1] + ".gif";
            string savepath = textBoxImgPath.Text;
            string savename = "";
savename = fonttype[0] + "-" + fontword[0] + "-" + fontA[0];
            string imgType = imgUrl.Substring (imgUrl.
LastIndexOf("."));
    //如果同一作者有多个字体图片,则后面的序号依次增加
            for (int j = 1; j < 100; j++)
            {
if (! File.Exists(savepath + savename + "-" + j + imgType))
                {
                    savename += "-" + j + imgType;
                    break;
                }
            }
            //抓取图片并保存
            GetImg(imgUrl, savepath + savename);
            progressBar1.Value++;
            fontnum = i + 1;
}
progressBar1.Value = 0;
labelTS.Text = " 提示：抓取完成，共抓取" +
fontnum + "个"" + textBoxFont.Text + ""字";
        }
    }
```

下载字体图片文件的自定义函数代码如下：

```
//从指定位置抓取图片,保存到指定的路径文件名中
    private void GetImg (string imgUrl,String
savepathname)
    {
        //抓取请求;
        System.Net.HttpWebRequest request =
(System.Net.HttpWebRequest)System.Net.WebRequest.
Create(imgUrl);
        request.UserAgent = "Mozilla/6.0 (MSIE 6.0;
Windows NT 5.1; Natas.Robot)";
        request.Timeout = 3000;

        try
        {
            //返回抓取的图片,保存到流中
            System.Net.WebResponse response =
request.GetResponse();
    Stream stream = response.GetResponseStream();
            //创建文件流
            FileStream fso = new FileStream
(savepathname, FileMode.Create);
            //从图片流中按 K 字节读取,并存入文件流
            int imgLong = (int)response.ContentLength;
            byte[] arrayByte = new byte[1024];
            int l = 0;
            while (l < imgLong)
            {
                int i = stream.Read(arrayByte, 0, 1024);
                fso.Write(arrayByte, 0, i);
                l += i;
            }
            fso.Close();
            stream.Close();
            response.Close();
        }
        catch
        {
MessageBox.Show("从网络抓取字体图片出错");
        }
    }
```

4.3 处理字体图片

从网络上得到的字体图片和通过照相扫描得来的字体图片，由于来源多样，导致字体图片格式不一，大小不一，颜色也不一，在使用时很不方便，特别是显示在米字格中里，会造成字体大小不一，起不到米字格的定位作用。因此要对字体图片进行处理。

对字体图片的处理包括裁剪周围白边，改变字体颜色，对字体大小进行缩放等操作。

可以对得到的单个字体图片进行处理，也可以对得到的多个字体文件进行批量处理。

4.3.1 得到一个像素所占的字节数

不同的图像格式，一个像素所占的字节数是不同的，要对图片进行处理，需要知道图片的一个像素所占的字节数，以便在处理时进行判断。得到一个像素所占的字节数的代码如下：

```
private int getByteNum(Bitmap srcbmp)
{
    int pixellen = 1;//一个像素点的字节长度,不同的图像
//格式,每个像素点有不同的字节数。

    //根据图像格式,判断一个像素所占的字节数
        if (srcbmp.PixelFormat == PixelFormat.
Format8bppIndexed)
        pixellen = 1;
      else if (srcbmp.PixelFormat == PixelFormat.
Format16bppGrayScale)
        pixellen = 2;
      else if (srcbmp.PixelFormat == PixelFormat.
Format24bppRgb)
        pixellen = 3;
      else if (srcbmp.PixelFormat == PixelFormat.
Format32bppRgb)
        pixellen = 4;
      else if (srcbmp.PixelFormat == PixelFormat.
Format32bppArgb)
        pixellen = 4;
    return pixellen;
}
```

4.3.2 裁剪字体图片空白边缘

将字体图片读入内存，逐字节自动查找字体图片的实际像素边缘，然后裁剪图片，将字体图片周围的空白边缘去掉。代码如下：

```
        private Bitmap CutBlank(Bitmap srcbmp)
        {
            int pixellen = 1;//一个像素点的字节长度,不同的
//图像格式,每个像素点有不同的字节数。
            //定义字体实际像素的矩形
            Rectangle cutRect = new Rectangle (0,0,
srcbmp.Width, srcbmp.Height);
            //读入原图的 BitmapData
            System.Drawing.Imaging.BitmapData bmpData
= srcbmp.LockBits (new Rectangle (0, 0, srcbmp.Width,
srcbmp.Height), System.Drawing.Imaging.ImageLockMode.
ReadWrite,srcbmp.PixelFormat);
            //根据图像格式,判断一个像素所占的字节数
            pixellen = getByteNum(srcbmp);
            // 得到图片第一行像素的地址.
            IntPtr ptr = bmpData.Scan0;
            //使用一维数组法,在内存中操作,查找字体边缘,
//速度介于 Bitmap 中 Getpixel 和指针法之间。
            // 声明一个装载位图的一维数组.
            int bytes = srcbmp.Width * srcbmp.Height *
pixellen;// 数组的长度.
            byte[] rgbValues = new byte[bytes];
            bool iffind = false;//标记是否找到边缘
            int pixelposition;//最靠边的像素位置
            // 把位图的 RGB 值拷入此数组.
            System.Runtime.InteropServices.Marshal.Copy
(ptr, rgbValues, 0, bytes);
            //从上边开始,向下一次移 2 个像素画横线,直到
//碰到字体实际像素为止,此处即为字体上边缘 Y
            for (int i = 0; i < srcbmp.Height; i += 2)
            {
                for (int j = 0; j < srcbmp.Width; j += 1)
                {
                pixelposition = (i * srcbmp.Width + j) * pixellen;
                    if (rgbValues[pixelposition] > 0)
                    {
                        //上边留一个像素的白边,所以-2
                        if(i>1) cutRect.Y = i-2;
                        iffind = true;
                        break;
                    }
                }
                if (iffind)
                {
                    iffind = false;
                    break;
                }
            }
            //从下边开始,向上一次移 2 个像素画横线,直到
//碰到字体为止,再减去上边 Y,得到字体高度
            for (int i = srcbmp.Height-1; i > 0; i -= 2)
            {
                for (int j = 0; j < srcbmp.Width; j += 1)
                {
                pixelposition = ((i-2) * srcbmp.Width + j) * pixellen;
                    if (rgbValues[pixelposition] > 0)
                    {
                if (i > 1) cutRect.Height = i-cutRect.Y+2 ;
                        iffind = true;
                        break;
                    }
                }
                if (iffind)
                {
                    iffind = false;
```

```
                    break;
                }
            }
            //从左边开始,向右一次移 2 个像素画竖线,直到
//碰到字体为止,此处即为字体左边缘 X
            for (int i = 0; i < srcbmp.Width; i += 2)
            {
                for (int j = 0; j < srcbmp.Height; j++)
                {
  pixelposition=(j * srcbmp.Width + i)*pixellen;
                    if (rgbValues[pixelposition] > 0)
                    {
                        //左边留一个像素的白边,所以-2
                        if(i>1) cutRect.X = i-2;
                        iffind = true;
                        break;
                    }
                }
                if (iffind)
                {
                    iffind = false;
                    break;
                }
            }
    //从右边开始,向左一次移 2 个像素画竖线,直到碰到字体
//为止,此处即为字体右边缘,再减去左边 X,得到字体宽度
            for (int i = srcbmp.Width-1; i > 0 ; i-=2)
            {
                for (int j = 0; j < srcbmp.Height; j++)
                {
        pixelposition=(j * srcbmp.Width + i)*pixellen;
                    if (rgbValues[pixelposition] > 0)
                    {
                        //此处+4 是因为,在图片外留一个像素
//的白边,所以,左边向左移了 2 个像素,右边向右移了 2 个像素,
//因此,宽度是+4
                        cutRect.Width = i-cutRect.X+4;
                        iffind = true;
                        break;
                    }
                }
                if (iffind)
                {
                    iffind = false;
                    break;
                }
            }
            // Unlock the bits.
            srcbmp.UnlockBits(bmpData);
            //建立新 bitmap
            Bitmap newbmp = new Bitmap(cutRect.Width,
cutRect.Height);
                //创建新 bitmap 的画板
            Graphics g = Graphics.FromImage (newbmp);
//定义字体文件的透明色
                        ImageAttributes fontAttr = new
ImageAttributes();//定义在 System.Drawing.Imaging
            fontAttr.SetColorKey (srcbmp.GetPixel (2, 2),
srcbmp.GetPixel(2, 2));
            //在画板上画裁剪过的字体图片
            g.DrawImage(srcbmp,//叠加图
                new Rectangle(0, 0, cutRect.Width, cutRect.Height),
//要叠加到背景图的位置,尺寸
                cutRect.X, cutRect.Y, cutRect.Width, cutRect.Height,
//要叠加的尺寸
            GraphicsUnit.Pixel, fontAttr);      //透明阀值
            //返回裁剪掉空白边的字体实际像素的 bitmap
            return newbmp;
        }
```

4.3.3 改变字体图片颜色

从不同来源得到的字体图片颜色不尽相同，为美观起见，将字体图片上的实际像素点统一改变为黑色，并且在改变过程中，将字体边缘的淡化颜色也改为黑色，起到锐化字体边缘的作用。

改变字体图片颜色代码如下：

```
        private Bitmap turnColor(Bitmap bmp)
        {
            int pixellen = 1;//一个像素点的字节长度,不同的
//图像格式,每个像素点有不同的字节数。
            //根据图像格式,判断一个像素所占的字节数
            pixellen = getByteNum(bmp);
            // Lock the bitmap's bits.
            Rectangle rect = new Rectangle (0, 0, bmp.
Width, bmp.Height);
    System.Drawing.Imaging.BitmapData bmpData =bmp.
LockBits (rect, System.Drawing.Imaging.ImageLockMode.
ReadWrite, bmp.PixelFormat);

            // Get the address of the first line.
            IntPtr ptr = bmpData.Scan0;
            //使用一维数组法,在内存中操作,查找字体边缘,
//速度介于 Bitmap 中 Getpixel 和指针法之间。
            // 声明一个装载位图的一维数组.
            int bytes = bmpData.Stride * bmp.Height;
            byte[] rgbValues = new byte[bytes];
            // Copy the RGB values into the array.
             System.Runtime.InteropServices.Marshal.Copy
(ptr, rgbValues, 0, bytes);
// Set every third value to 255. A 24bpp bitmap will look red.
  for  (int counter = 0; counter < rgbValues.Length; counter
+= pixellen)
                {
                    if (rgbValues[counter] > 0)
```

```
            {
             //四个字节排序为 bgra,所以把前三个置为同
//样的 8,即为黑色,第 4 个为 A,置为 255,则没有透明化,字体
//边缘清晰。
                rgbValues[counter] =rgbValues[counter +
1] = rgbValues[counter + 2] =  8;
                rgbValues[counter + 3] = 255;
            }
        }
        // Copy the RGB values back to the bitmap
         System.Runtime.InteropServices.Marshal.Copy
(rgbValues, 0, ptr, bytes);
        // Unlock the bits.
        bmp.UnlockBits(bmpData);
        return bmp;
    }
```

4.3.4 改变字体图片大小

从不同来源得到的字体图片大小不一，有些可能过大，存储时占用空间过多，有些可能过小，放大后边缘粗糙，锯齿过大，因此，将图片缩放为统一大小，并在缩放时对边缘平滑化，方便对字体图片的存储和显示。

```
    private Bitmap resizeBmp(Bitmap bmp)
    {
  int fontwidth, fontheight;//字体显示时用的宽度和高度
        //对字体进行缩放,计算缩放后字体的宽和高。
        //如果是长形字体
        if (bmp.Width < bmp.Height)
        {
            fontheight = 600;
   fontwidth = (int)((bmp.Width * fontheight / bmp.Height));
        }
        else//如果是宽形字体
        {
            fontwidth = 600;
fontheight = (int)((bmp.Height * fontwidth / bmp.Width));
        }
        //设定字体图片写入模式为平滑模式
  Bitmap returnbmp = new Bitmap(fontwidth, fontheight);
        System.Drawing.Graphics G = System.
Drawing.Graphics.FromImage(returnbmp);
    G.SmoothingMode = System.Drawing.Drawing2D.
SmoothingMode.HighQuality;
        //定义字体文件的透明色
    ImageAttributes fontAttr = new ImageAttributes();//定义
//在 System.Drawing.Imaging
        fontAttr.SetColorKey (bmp.GetPixel (2, 2), bmp.
GetPixel(2, 2));
        //在画板上画裁剪过的字体图片
        G.DrawImage(bmp,//叠加图
    new Rectangle(0, 0, fontwidth, fontheight),//要叠加到背
//景图的位置,尺寸
    0, 0, bmp.Width, bmp.Height,//要叠加的尺寸
    GraphicsUnit.Pixel, fontAttr);  //透明阀值
        return returnbmp;
    }
```

4.3.5 字体图片综合处理及存储

对特定一个字体图片，其处理包括裁剪白边、改变颜色、改变大小、重命名、存储到指定位置等一系列工作，实现代码如下：

```
    private void TreatFont(string filepathname)
    {
        //读取字体图片
        Bitmap srcbmp = new Bitmap(filepathname);
  Bitmap fontbmp;
        //查找实际像素的矩形边框,裁剪周边的空白边
        fontbmp = CutBlank(srcbmp);
        srcbmp.Dispose();
        //改变字体图片大小
        fontbmp = resizeBmp(fontbmp);
        //改变字体颜色为黑色
        fontbmp = turnColor(fontbmp);
         //以.分隔字体文件的路径文件名,取得.前面的部
//分,以便改变扩展名后以 png 格式存储
    string[] fontA = filepathname.Split(new char[] { '.' });
        string newfilepathname=fontA[0] + ".png";
        //如果新文件名存在,则删除,并重新存储
        if (File.Exists(newfilepathname))
        {
            File.Delete(newfilepathname);
        }
        //将原路径文件名拆分为路径和文件名,根据文件
//名中的书法文字,确定新目录名,如果新目录不存在,则创建新
//目录。并把处理后的图片存入新目录内。
    //得到最后一个反斜杠的位置,并以此拆分路径和文件名。
        int tmppos=newfilepathname.LastIndexOf('\\');
        string tmppath = newfilepathname.Substring(0,
tmppos+1);//原路径
         string tmpname = newfilepathname.Substring
(tmppos+1);//原文件名
        //以短横杠拆分文件名,取得其中的书法文字
    string[] tmpnameA = tmpname.Split(new char[] { '-' });
    //新路径
     string newpath = tmppath + tmpnameA[1] + '\\';
        //如果新目录不存在,则创建新目录
        if (! Directory.Exists(newpath))
        {
            Directory.CreateDirectory(newpath);
        }
        //把处理好的图片以 png 格式存储到新目录内。
 fontbmp.Save(newpath+tmpname,ImageFormat.Png);
srcbmp.Dispose();
        fontbmp.Dispose();
```

```
            //删除原文件
            if (File.Exists(filepathname))
            {
                File.Delete(filepathname);
            }
        }
```

如果需要同时处理多个字体图片，则对该批字体图片进行批量处理，代码如下：

```
//点击处理按钮时,查找指定目录下的字体文件,并处理
        private void buttonCutFlag_Click (object sender,
EventArgs e)
        {
            //进行处理前,先查找字体,返回保存字体路径文
//件名的数组,然后拆分为数组。
                string [] fontpathnameA = FindFile
(textBoxImgPath.Text).Split(new char[] { ';' });
            labelTS.Text = "提示:共找到" + fontpathnameA.
Length + "个字体文件,正在处理中……";
    //以数组长度,即要处理字体的个数对进度条赋值。
            progressBar1.Minimum = 0;
    progressBar1.Maximum = fontpathnameA.Length;
  //遍历数组,对字体逐一进行处理,以进度条显示处理进度
            for (int i = 0; i < fontpathnameA.Length; i++)
            {
                TreatFont(fontpathnameA[i]);
                progressBar1.PerformStep();
            }
            //处理完成后,显示处理的字体个数。
            labelTS.Text = " 提示：共处理字体个数为" +
fontpathnameA.Length;
            progressBar1.Value = 0;
        }
```

5 结语

使用 C# 编程制作了一个实用的书法字典程序。本书法字典程序的字库以图片形式保存，以文件夹形式存储，存储历代书法名家的字体图片，通过作者和不同字体为条件对字库进行检索，可以找到指定的字体并显示。

由于字体是以书法名家的原字体图片形式保存，字体完整保留了书法名家的字体真迹和神韵。在字体显示时，对字体进行放大，并按比例放入米字格，方便书法爱好者读贴和临摹。并且由于字库以文件夹形式存储，具有易于扩充，便于检索的特点、同时本程序还提供了字体图片的批量提取、裁剪、改变颜色，改变大小的处理方法，极大方便了书法爱好者对字库的自行扩充。

本程序对书法爱好者是一个易于使用，便于维护的极佳工具和学习助手。

（收稿日期：2012-12-12）

（上接第 29 页）

```
        else
        {
            return 0;
        }
}
```

6 模拟测试

输入 5 个初始记录，并分别测试文件记录的保存、查找和删除功能，程序运行稳定。测试过程如图 5 所示。

图 5 文件系统的模拟测试

7 结语

在哈希结构模拟文件系统的过程中，Hash Table 结构中的各个域均设置为整型，整个哈希表长度为 MAXSIZE，在文件开头处宏定义为 8，初始化数组中含 ARRAY=5 个数据，这样的设置均是为了方便编程的实现。在实际的文件系统中，文件拥有的属性要复杂很多，包括文件创建时间、修改时间、文件主、文件类型等，本例旨在分析哈希算法在保存、查找和删除记录时的流程和思想，以供广大读者和编程人员参考。

参考文献

[1] 汤小丹，梁红兵，哲凤屏，等. 计算机操作系统 [M] . 3 版. 西安：西安电子科技大学出版社，2007.

[2] 严蔚敏，吴伟民. 数据结构（C 语言版） [M] . 北京：清华大学出版社，2007.

（收稿日期：2012-09-04）

巧用 SSMS 解决 Excel 筛选难题

李 斌

摘 要：介绍利用 SSMS 解决 Excel 多表格筛选问题的方案。

关键词：SSMS 工具；Excel 软件；筛选

1 问题提出

Excel 是日常办公的必备工具，其中表格筛选是比较耗费精力的一项工作，尤其是多表格筛选问题，如表 1、表 2 两个实例所示。

表 1 例一

字段 1	字段 2	字段 3
Xxx	Yyy	Zzz
Aaa	Bbb	Ccc
…	…	…

表 2 例二

字段 4	字段 5
Zzz	60
Lll	-15
…	…

现在的要求是对表 1 进行筛选，当表 1 字段 3 的内容存在于表 2 字段 4 之中，且同时表 2 该行字段 5>0 时保留该行，对其余行则舍弃。这个问题直接使用 Excel 现有功能较难解决，如果要在 Excel 应用程序中完成此要求，可能需要借助 VBA (visual basic for application) 进行一些程序设计工作才行，如果对 Excel 编程模型及 VBA 语言不够熟悉，相关工作将比较困难，下面笔者介绍通过 SSMS (SQL Server management studio) 来解决此问题，读者只需具备简单的 SQL 数据库知识即可，而几乎不需要编程。

2 SSMS 简介

SSMS 是微软公司为 SQL Server 提供的图形化管理工具，由于通过命令行操纵 SQL Server 数据库较为复杂，而使用图形化工具则相对容易得多，因此利用好 SSMS 将大幅简化使用 SQL Server 的操作，这给我们利用 SQL Server 进行数据处理带来了方便。

3 解决方案

为了解决这个筛选难题，首先将两个 Excel 表导入到 SQL Server 数据库 (可新建一个空数据库) 中作为数据表，如图 1 所示，导入过程依照提示进行，数据源处请选择 Microsoft Excel 如图 2 所示，如果表格带有标题行，则选中“首行包含列名称”，随后 SSMS 会自动连接数据库，并生成数据导入的 SQL 命令并执行导入过程，所有步骤都是图形化界面下的操作，非常简便。

图 1 导入数据

图 2 选择数据源

（下转第 48 页）

基于JSON的柔性数据维护平台实现

汪永松

摘　要：通过两条主线对实现思路进行阐述：第一条主线是基于JSON交互搭建多层B/S应用架构，其实现了数据的集成；另外一条主线是数据维护平台的实现模式以及其柔性体现。这样不仅可以了解多层B/S应用架构的搭建过程，而且还能对数据维护平台的设计模式形成一定的认识。

关键词：JSON格式；B/S多层架构；Ajax框架；数据维护

1　概述

对于数据的维护，一般应用是对数据表记录行的（新）增、删（除）、（修）改、查（询）。而文中的数据维护，包含两个方面：对数据表记录行和模式信息的维护。对于记录行的维护与一般的应用相同；而对于数据表的模式信息的维护则可以理解为对表列信息的维护，即对表列的增、删、改、查。

按照设计，通过对数据表记录行和模式信息这两个方面的维护，管理员可以对数据表内容进行任意地扩展。不仅如此，维护平台对数据表模式信息的获取采用柔性的方式，自动适应模式信息的更新，并自动生成相应维护界面。

此外，该平台采用三层架构（数据层、数据维护服务层和Web前端）模式，数据维护的具体操作都使用服务接口的方式，这些设计对于增强平台的扩展性是显而易见的。

1.1　数据维护平台实例

数据维护包含两个方面：对数据表记录行和模式信息的维护，图1中是维护数据表记录行的主界面；而图2是对维护数据表模式信息的主界面。

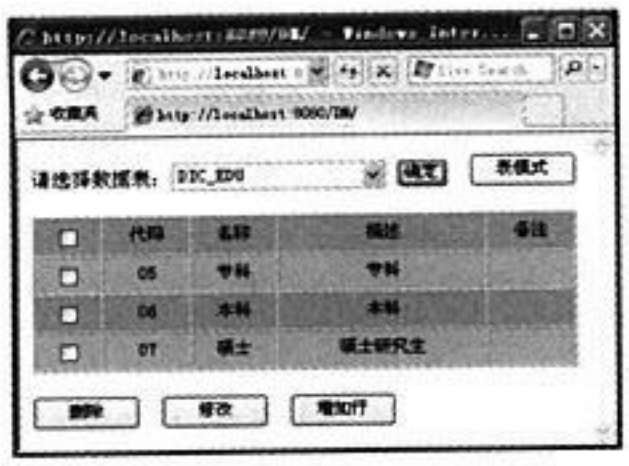


图1　维护数据表记录行

图1中的表格内容行实际上是对表查询得到的结果集，而记录行的删除、修改和新增则是通过对应的功能按钮的方式进行提供。其中，删除和修改操作是需要先选定记录行，所以在各行之前添加了一个用于选取的复选框；有所不同的是：删除操作可以选取多条记录，而修改操作只能选取一条记录。

需要注意的是，图1中表头部分是各个列的名称，该信息无法从查询结果中获取，而是需要从数据表模式信息中获取。

图2与图1的内容框架基本相同，其主要差异在于表头内容，对于记录维护界面而言，不同表的表头是不相同的（表不同，其列也不同）；而对于模式维护界面，不同表的表头都是相同的（表不同，但其列的信息项是相同的）。其中表列的信息项设定为：名称、类型、大小（长度）、小数点位数（对于数组类型的列）、是否可为空、是否为主键和备注，图1中表头的内容即为各列的备注项。

图2　维护数据表模式

1.2　平台架构分析

该数据维护平台采用三层架构：数据层、数据维护服务层和Web前端。其中，数据维护功能都是以服务接口的方式提供的，表1中即为数据维护功能和对应服务接口的说明。

表1　数据维护功能与服务接口说明

序号	一级功能	二级功能	服务接口	说明
1	记录维护	查询	getTblRows	获取表记录行
			getTblPk	获取表主键信息
2		删除	delTblRows	删除表记录行
3		修改	updTblRow	更新表记录行
4		新增	addTblRow	新增表记录行

序号	一级功能	二级功能	服务接口	说明
5	模式维护	查询	getTblSpec	获取表模式信息
			getTblPk	获取表主键信息
6		删除	delTblCols	删除表列
7		修改	updTblCo	更新表列
8		新增	addTblCol	新增表列

表 1 中的服务接口用于执行后台操作，本身不提供用户界面，其通过 Web 前端进行调用。

Web 前端通过与用户进行交互，获取相应的参数信息，当用户确定操作请求时，Web 前端调用对应的数据维护服务接口并传入参数，当功能执行完毕，还会将服务接口的执行结果反馈给用户。在数据维护平台的设计中，执行结果和数据集都通过 JSON 格式进行传输，数据维护服务接口采用 Ajax 的方式调用（包括参数传递）。

图 3 是该数据维护平台的主要功能流程。

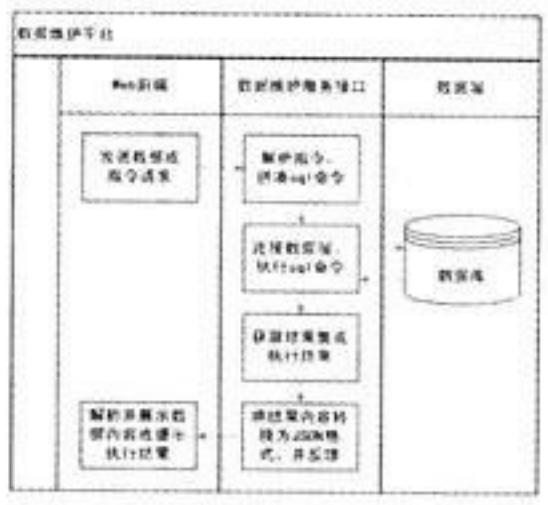


图 3 数据维护平台业务流程

1.3 实现要点

根据对数据维护平台的功能架构分析，平台的实现要点应该包括两个部分：数据维护服务接口定义和数据维护的柔性。

1.3.1 数据维护服务接口定义

数据维护平台采用 Ajax 方式从服务接口获取数据，交互的结果传输统一使用 JSON 格式。输出内容主要包括 4 种：数据表模式、数据表主键、结果集（查询操作）和执行结果（插入、删除和更新操作）。

（1）模式信息的输出格式

[{"1":"Name" ,"2":"Type","3":"Precision","4":"Scale","5":"isNullable","6":"isPk","7":"Remarks"},

{"1":"<第 1 列列名>" ,"2":"<第 1 列类型>","3":"<第 1 列大小>","4":"<第 1 列小数位数>","5":"<第 1 列是否为空>","6":"<第 1 列是否为主键>","7":"<第 1 列备注信息>"},

{"1":"<第 2 列列名>" ,"2":"<第 2 列类型>","3":"<第 2 列大小>","4":"<第 2 列小数位数>","5":"<第 2 列是否为空>","6":"<第 2 列是否为主键>","7":"<第 2 列备注信息>"},

…

]

（2）主键信息的输出格式

[{"1":"<主键列名>";"2":"<主键列类型>";"3":"<主键列序号>";}]

（3）结果集的输出格式

[{"i":"第 i 列列名","j":"第 j 列列名",……},

{"i":"结果集第 1 行第 i 列列值","j":"结果集第 1 行第 j 列列值",……},

{"i":"结果集第 2 行第 i 列列值","j":"结果集第 2 行第 j 列列值",……},

…

]

（4）执行结果的输出格式

[{RESULT:<执行结果>}]

一般为了增强代码的可读性，数据维护服务接口定义中的数据项在前后端都需要进行定义，其中前端以 JavaScript 的语法进行定义，后端以 Java 的语法进行定义。代码 1 即为前端对数据项定义：

代码 1 数据项定义

文件名：tbl_schema.js

```
//列信息项
Constants.Name = 1;
Constants.Type = 2;
Constants.Precision = 3;
Constants.Scale = 4;
Constants.isNullable = 5;
Constants.isPk = 6;
Constants.Remarks = 7;
//主键信息项
Constants.PkName = 1;
Constants.PkType = 2;
Constants.PkSeq = 3;
//结果信息项
Constants.Result = "RESULT";
Constants.ResultOk = "SUCCESS";
```

1.3.2 数据维护的柔性

（1）数据加载的柔性：在加载数据表的记录行之前会获取模式和主键信息，并使用这些信息自动调整记录行的显示效果和添加处理标识。例如：使用列中文备注信息替换英文列名进行显示，有助于提高界面的友好程度；使用每条记录行的主键值来标识每行记录，为删除、修改操作做好铺垫。

（2）记录行编辑的柔性：JSP 页面会根据数据表的模式和主键信息自动生成新增或更新界面的控件并为各控件设置区分标识；更新界面会自动检测数据项的改动状况，做到只更新变动的内容。

（3）维护功能实现的柔性：使用服务接口的方式提供对应的数据维护功能，采用标准的 JSON 格式进行交互，有利于架构层次和功能结果的调整和扩展。

此外，通过数据表的模式和主键信息，还可以平衡前后端

的数据处理工作。例如：对查询值、更新值和插入值的组织需要参考列类型信息（例如：字符串类型的列值需要加引号、日期类型的列值需要通过函数进行转换），而这些工作也可以在页面前端进行，避免了后台的集中处理。

2 设计过程

2.1 设计思路

限于篇幅，只介绍几个较为典型的类的设计。

2.1.1 数据表模式信息服务接口—getTblSpec.ws

数据表模式信息服务接口的输入是目标表名，通过 JDBC API 获取该表各列的信息（名称、类型、大小、是否可以为空、是否主键、备注信息），并将该信息转换成 JSON 格式进行返回。

注意：为了区分普通的 JSP 单元，服务接口采用“ws”的后缀，但其实质还是 JSP 单元，只是其不会直接输出内容到前端界面。为了保证 ws 单元能够执行，需要在应用程序服务器（本文中为 Tomcat）的“web.xml”文件中进行映射设置，如代码 2 所示：

代码 2　对服务接口单元进行映射

文件名：web.xml

```
<servlet-mapping>
    <servlet-name>jsp</servlet-name>
    <url-pattern>*.ws</url-pattern>
</servlet-mapping>
```

2.1.2 数据表主键信息服务接口—getTblPk.ws

实际上数据表模式信息中已经包含了主键信息，但是为了获取为主键的列，需要对所有列进行遍历。所以，为了避免该遍历过程，特提供了该接口。

该接口的输入也是目标表名，通过 JDBC API 获取该表的主键信息及主键列信息（列名、类型和列序号），并将该信息转换成 JSON 格式进行返回。

2.1.3 数据维护主页—index.htm

数据维护主页采用 jQuery 库的 Ajax 方法调用数据服务接口（数据表信息、表模式信息、主键信息、表数据集信息、删除记录行接口），然后对反馈的结果进行解析并展示或提示。

由于服务接口的调用过程存在先后的制约（例如：表模式信息的获取必须在数据集获取之前），为了保证调用过程的串行，其中服务接口的调用采用嵌套，即在上一个接口调用的回调函数中进行下一个调用。

2.1.4 记录行新增页—add_tbl_row.jsp

记录行新增页从数据维护主页进行跳转，输入参数为目标表名，其会调用数据表模式和主键信息服务接口，自动生成数据行新增界面（如图 4 所示）；当提交数据库时，其又会调用新增表记录行的服务接口并返回执行结果。

2.1.5 修改表列页—edit_tbl_col.jsp

修改表列页从数据模式维护主页进行跳转，输入参数为目标表名和目标列名，其会调用数据表模式服务接口，获取表列信息并填充网页编辑控件，同时通过隐藏控件备份原值；当提交修改时，其会判断修改项，并调用更新表列的服务接口并返回执行结果。

对于表记录行的维护，其最终都是通过执行 SQL 命令来实现；而对于表数据模式的维护，除了执行 SQL 命令外，还有使用 JDBC API。有关实现数据维护服务接口所用到的 SQL 命令或 API 如表 2 所示。

表 2　数据维护服务接口的实现方式

序号	服务接口	实现方式
1	获取表记录行	select * from <表名>
2	获取表主键信息	JDBC 类“DatabaseMetaData”的“getPrimaryKeys”方法
3	删除表记录行	delete from <表名> where <主键>=<主键值>
4	更新表记录行	update <表名> set <列名>=<列值> where <主键>=<主键值>
5	新增表记录行	insert into <表名> (列名集合) values (列值集合)
6	获取表模式信息	JDBC 类“DatabaseMetaData”的“getColumns”方法
7	删除表列	alter table <表名> drop column <列名>
8	更新表列（列名）	alter table<表名> rename column <旧列名> to <新列名>
9	更新表列（类型）	alter table <表名> modify (<列名> <类型> [(大小 [,小数点数])])
10	更新表列（是否为空）	alter table <表名> modify (<列名> <NULL|NOT NULL>)
11	新增表列	alter table <表名> add (<列名> <类型> [(大小 [,小数点数])])
12	设置或更新列备注	comment on column <表名>.<列名> is <备注>

需要注意的是：表 2 中的 SQL 命令可能并不适用所有类型的数据库，这里使用的是 Oracle 数据库（10gR2）。

2.2 数据库设计

案例中使用的示例数据表是学历字典表（DIC_EDU），其结构定义如表 3 所示。

表 3　学历字典表

字段名	类型	约束	默认值	备注
DM	VARCHAR2 (32)	主键		代码
MC	VARCHAR2 (50)			名称
MS	VARCHAR2 (100)			描述

3 代码详解

3.1 数据维护服务接口

3.1.1 数据表模式信息

数据表模式信息服务接口主要工作是获取模式信息并将内

容返回给请求端，其关键处理如代码 3 所示：

代码 3　数据表模式信息服务接口

```
文件名:getTblSpec.ws
//初始化 JSON 数组对象(存放 JSON 对象)
JSONArray array = new JSONArray();
try { //连接数据库,获取指定数据表的结果集
      conn = DriverManager.getConnection(url, prop);
      DatabaseMetaData dbms = conn.getMetaData();
rs1 = dbms.getPrimaryKeys(null, dbms.getUserName(), tbl);
      //主键列名
HashSet<String> pkCols = new HashSet<String>();
//获取主键列名
while(rs1.next()) {
      pkCols.add(rs1.getString("COLUMN_NAME") );
}
rs1.close();
rs1 = null;
//表头部分的模式信息
JSONObject header = new JSONObject();
header.put("1", IConfig.NAME);
header.put("2", IConfig.TYPE);
header.put("3", IConfig.PRECISION);
header.put("4", IConfig.SCALE);
header.put("5", IConfig.ISNULLABLE);
header.put("6", IConfig.ISPK);
      header.put("7", IConfig.REMARKS);
      array.put(header);
      //获取表列信息
rs2 = dbms.getColumns(null,dbms.getUserName(), tbl, col);
      while(rs2.next()) {
            JSONObject colSpec = new JSONObject();
            colSpec.put("1", rs2.getString("COLUMN_NAME"));
            colSpec.put("2", rs2.getString("TYPE_NAME"));
            colSpec.put("3", rs2.getInt("COLUMN_SIZE"));
            colSpec.put("4", rs2.getInt("DECIMAL_DIGITS"));
            colSpec.put("5", (rs2.getInt("NULLABLE")==1) );
            colSpec.put ("6", pkCols.contains (rs2.getString ("
COLUMN_NAME")) );
            colSpec.put("7", rs2.getString("REMARKS"));
            array.put(colSpec);
      }
```

代码 3 中，使用了 JDBC 类“DatabaseMetaData”的“getPrimaryKeys”和“getColumns”方法来获取主键和各列的信息，并按照既定格式添加到 JSON 容器中。

其中获取主键信息是为了对表列信息是否为主键进行判断；此外，为了获取列的备注信息，还需要设置允许获取列备注信息，如代码 4 所示：

代码 4　设置允许获取列备注信息

```
文件名:getTblSpec.ws
Properties prop = new Properties();
prop.setProperty("user", user);
prop.setProperty("password", password);
prop.setProperty("remarks", "true");
```

最后，将 JSON 容器中的内容按设计规范定义的格式返回给调用段，如代码 5 所示：

代码 5　以 JSON 格式回复请求

```
文件名:getTblSpec.ws
//以 JSON 格式回复请求端
response.setContentType("text/plain");
response.setCharacterEncoding("utf-8");
PrintWriter writer = response.getWriter();
writer.write(array.toString());
writer.flush();
writer.close();
```

3.1.2 数据表主键信息

数据表主键信息服务接口主要工作是获取主键信息并将内容返回给请求端，其关键处理如代码 6 所示：

代码 6　数据表主键信息服务接口

```
文件名:getTblPk.ws
try { //连接数据库,获取当前用户所有的数据表
conn = DriverManager.getConnection(url, user, password);
      DatabaseMetaData dbms = conn.getMetaData();
      //获取主键信息
      String pkName="";
      int pkSeq=-1;
rs1 = dbms.getPrimaryKeys(null, dbms.getUserName(), tbl);
      while(rs1.next()) {
            pkName = rs1.getString("COLUMN_NAME");
            pkSeq = rs1.getInt("KEY_SEQ");
      }
      rs1.close();
      rs1 = null;
      //获取主键列信息
      rs2 = dbms.getColumns(null,dbms.getUserName(), tbl,
pkName);
      while(rs2.next()) {
            JSONObject col = new JSONObject();
            col.put(IConfig.PK_NAME, pkName);
      col.put(IConfig.PK_TYPE, rs2.getString("TYPE_NAME") );
            col.put(IConfig.PK_SEQ, pkSeq);
            array.put(col);
      }
```

代码 6 与代码 3 的代码内容都存在获取主键信息和列信息，但在目的上存在差异：代码 3 中获取主键信息是为了判断各列是否为主键，而代码 6 中是为了进一步获取主键列的信息；代码 3 中获取的是所有列的详细信息，而代码 6 中仅仅为了获取主键列的类型信息。

主键信息主要用于获取主键列值并用其标识各记录行，类型信息主要用于指导列值的包装（例如：字符串类型的列值必

须添加引号等)。

3.1.3 删除表列

删除表列服务接口主要工作是解析请求的列名集合，再根据列名生成相应的 SQL 命令并执行之，将执行结果反馈给请求端。其关键处理如代码 7 所示：

代码 7　删除表列服务接口

文件名:delTblCols.ws

```
try { //连接数据库,获取指定数据表的结果集
conn = DriverManager.getConnection(url, user, password);
                    stat = conn.createStatement (ResultSet.
TYPE_SCROLL_SENSITIVE,
                        ResultSet.CONCUR_READ_ONLY);
    //拼凑删除表列的 SQL 命令并执行
    for(int i = 0; i < cols2.length; ++i) {
final String sql= "alter table "+tbl+" drop column "+cols2[i];
        stat.executeUpdate(sql);
    }
    //关闭数据库连接
    ……
    //设置返回结果
    obj.put(IConfig.RESULT, IConfig.RESULT_OK);
} catch(Exception e) {
    obj.put(IConfig.RESULT, e.getMessage());
}
```

代码 7 中，服务接口根据表列数组拼凑成 SQL 命令并依次执行，最后将执行结果按照设计规范返回给调用段。其中，如果执行正常则将返回约定的正常值，否则直接将异常消息作为返回值。

3.1.4 删除表记录行

删除表记录行服务接口主要工作是解析请求的记录行标识(主键值)，再根据主键值生成相应的 SQL 命令并执行之，将执行结果反馈给请求端。其关键处理如代码 8 所示：

代码 8　删除表记录行服务接口

文件名:delTblRows.ws

```
try { //连接数据库,获取指定数据表的结果集
conn = DriverManager.getConnection(url, user, password);
                    stat = conn.createStatement (ResultSet.
TYPE_SCROLL_SENSITIVE,
                ResultSet.CONCUR_READ_ONLY);
    //拼凑删除表列的 SQL 命令并执行
    for(int i = 0; i < ids2.length; ++i) {
        final String sql= "delete from "+tbl+" where  ("+
pk+"="+ids2[i]+")";
        stat.executeUpdate(sql);
    }
    //关闭数据库连接
    ……
    //设置返回结果
    obj.put(IConfig.RESULT, IConfig.RESULT_OK);
} catch(Exception e) {
    obj.put(IConfig.RESULT, e.getMessage());
}
```

代码 8 中，服务接口根据记录行 ID 数组拼凑成 SQL 命令并依次执行，最后将执行结果按照设计规范返回给调用段。

3.2　数据维护页

3.2.1 代码框架

数据维护页的内容展示和功能调用都使用 jQuery 的 Ajax 方式，其框架如代码 9 所示：

代码 9　数据维护页代码框架

文件名:index.html

```
<meta  http –equiv = "Content –Type"  content = "text/html;
charset=gbk" />
<script type="text/javascript" src="jquery.js"></script>
<script type="text/javascript" src="tbl_schema.js"></script>
<link rel="stylesheet" type="text/css" href="public.css" />
<script type="text/javascript">
    $(document).ready(function() {
        ……//关键代码
    });
</script>
</head>
<body>
<div>请选择数据表:<span id="db_spec"></span>  <
button type="button" id="btn_ok">确定</button>
  <button  type = "button"  class = "func_btn"  id = "
btn_schema">表模式</button></div>
<p><div id="tbl_rows"></div></p>
<div id="bar"><button type="button" class="func_btn" id="
btn_del">删除</button>
 <button type="button" class="func_btn" id="btn_edit">
修改</button>
 <button type="button" class="func_btn" id="btn_add">
增加行</button></div>
```

代码 9 中，页面定义了两个 HTML 容器控件（“db_spec”和“tbl_rows”），前者用于获取当前用户的表名称列表，后者用于获取指定表的记录集。所有关键代码都在 jQuery 的页面准备完毕的回调函数中。

3.2.2 初始化

页面初始化内容包括获取数据库中表名称列表以及获取第一个表的记录集并显示，如代码 10 所示：

代码 10　数据维护页初始化

文件名:index.html

```
//获取数据库中表名称列表
$.getJSON("/DataServices/getDbTbls.ws",function(tbls) {
    //获取结果集列数
    var col_count = 0;
    $.each(tbls[0],function(i, col) {
        col_count++;
```

```
    });
    var html = "<select id=\"tbl\">";
    //遍历行
    $.each(tbls,function(row, col) {
        for(var i = 1; i <= col_count; ++i) {
html += ("<option value=\""+col[i]+"\">"+col[i]+"</option>");
        }
    });
    html += "</select>";
    //设置 HTML
    $('#db_spec').html(html);
    $('#btn_ok').click();
});
```

在代码 10 中，根据服务接口所反馈的信息生成了一段包含下拉选择框的 HTML 代码并设置给 HTML 容器控件，其效果如图 1 所示。当表名称列表获取完毕，则调用获取表记录集的方法来获取并显示表列中第一个表的记录集。代码 11 是获取表记录集的关键代码：

代码 11　获取表记录集

文件名：index.html

```
//显示指定表的内容
$('#btn_ok').click(function() {
    //获取主键列名及序号
    var uri1 = encodeURI ("/DataServices/getTblPk.ws?tbl="
+$('#tbl').val());
    $.getJSON(uri1,function(pks) {
        var pkName = pks[0][Constants.PkName];
        var pkSeq = pks[0][Constants.PkSeq];
        //获取数据表模式信息
        var uri2 = encodeURI ("/DataServices/getTblSpec.
ws?tbl="+$('#tbl').val());
        $.getJSON(uri2,function(colsSpec) {
            //获取表数据集
            getTblDataSet(pkName, pkSeq, colsSpec);
        });
    });
});
```

在代码 11 中，先获取表的主键信息，再根据主键信息获取数据集，其主要内容如代码 12 所示：

代码 12　获取表数据集

文件名：index.html

```
function getTblDataSet(pkName, pkSeq, colsSpec) {
    //获取数据集
var uri = encodeURI("/DataServices/getTblDataSet.ws?tbl="
                        +$('#tbl').val()+"&pk="+pkName);
    $.getJSON(uri, function(ds) {
  var html = " <table width =\"100% \" border =\"1\"
cellspacing=\"0\" cellpadding=\"0\">";
        //探测结果集列数
        var col_count = 0;
        $.each(ds[0],function(row, col) {
            col_count++;
        });
        $.each(ds,function(row, col) { //遍历记录行
            if(row==0) { //表头行
                html += ("<tr class=\"tr0\">");
                html += ("<td><input type=\"checkbox\" id=\"
chx_all\"></input></td>");
                //输出各列名或备注内容
                for(var i = 1; i <= col_count; ++i) {
        html += ("<td>"+((typeof (colsSpec [i][Constants.
Remarks])=="undefined")?
                colsSpec [i] [Constants.Name]:colsSpec [i]
[Constants.Remarks])+"</td>");
                }
            } else { //内容行
                html += ("<tr class=\"tr" + (row%2) + "\">");
 html += ("<td><input type=\"checkbox\" id=\"row_" +col
[pkSeq]+"\"></input></td>");
                //输出各列内容
                for(var i = 1; i <= col_count; ++i) {
        html += ("<td>"+((typeof (col [i])=="undefined")?"
 ":col[i])+"</td>");
                }
            }
            html += ("</tr>");
        });
        html += "</table>";
        //设置 HTML
        $('#tbl_rows').html(html);
    });
}
```

代码 12 中，主要内容是根据记录行的值内容生成 HTML 代码。为了提供对各行进行选择，在每行第一列设置了一个复选框，并通过将各行的主键值设置为复选框的 ID 来区分对各行的选择。此外为了增强界面的友好程度，记录行的各列尽可能采用备注信息作为表头，其效果如图 1 所示。

需要说明的是：使用各行的主键值来区分对各行的选择，主要是为删除和更新记录行做铺垫。

3.2.3 删除指定记录行

删除指定记录行的操作由功能按钮触发，该模块首先会根据行的选择获取主键值集合，再根据主键值集合来调用服务接口以实现记录行的删除，其主要内容如代码 13 所示：

代码 13　删除指定记录行

文件名：index.html

```
//删除指定行
$('#btn_del').click(function() {
    var ids = [];
    $(":checkbox[checked][id! ='chx_all']").each(function(){
        ids.push($(this).attr("id").replace("row_",""));
```

```
    });
    //获取主键列序号及名称
    var uri1 = encodeURI ("/DataServices/getTblPk.ws?tbl="
+$('#tbl').val());
    $.getJSON(uri1,function(pks) {
        var pkName = pks[0][Constants.PkName];
        var pkType = pks[0][Constants.PkType];
        //根据主键类型包装列值
        if( (pkType.indexOf("VARCHAR")! =-1) || (pkType.
indexOf("DATE")! =-1) ) {
            for(var i = 0; i < ids.length; ++i) {
                ids[i]=("\'"+ids[i]+"\'");
            }
        }
        //执行删除主键值为指定值的行
        var uri2 = encodeURI ("/DataServices/delTblRows.
ws?tbl="+
            $('#tbl').val()+"&pk="+pkName+"&ids="+ids);
        $.getJSON(uri2, function(result) {
            alert (getResultDesc ("删除行", result
[Constants.Result]));
            window.location.reload();
        });
    });
});
```

代码 12 中，首先是解析各行标识中的主键值，然后获取主键值信息，并根据主键值类型对主键值进行包装，最后根据包装后的主键值集合来调用删除记录行的服务接口以执行操作，并返回执行结果。

3.3 新增记录行页

3.3.1 代码框架

新增记录行页的界面内容是根据数据表的模式信息自动生成，所以其页面的主要内容是一个 HTML 容器，其代码框架如代码 14 所示：

代码 14 新增记录行页代码框架

文件名：add_tbl_row.jsp

```
<%
    final String tbl = request.getParameter("tbl");
%>
...
<script type="text/javascript">
    $(document).ready(function(){
        var colCount = 0;
        var colNames = new Array();
        var colTypes = new Array();
        …关键代码
});
</script>
<p>表【<%=tbl%>】新增行：</p>
<span id="content_view"></span>
<p><button type="button" id="btn_ok">确定</button> 
<button type="button" id="btn_cancel">取消</button></p>
```

代码 14 中，其页面包行一个 ID 为“content_view”的 HTML 容器控件，其关键代码也都在 jQuery 的回调函数中。

需要注意的是，在 JavaScript 代码中，使用两个全局的数组来装载各表列名和类型信息，和前面介绍删除记录行的模式一样，列名用来生成 SQL 语句，类型值用来包装列值。

3.3.2 初始化

新增记录行页的初始化工作主要是获取表模式信息，并动态生成输入界面，其关键处理如代码 15 所示：

代码 15 新增记录行页初始化

文件名：add_tbl_row.jsp

```
//获取数据表模式信息
$.getJSON ("/DataServices/getTblSpec.ws?tbl=<%=tbl%>",
function(colsSpec) {
    var html = "<table width =\"400\" border =\"1\"
cellspacing=\"0\" cellpadding=\"0\">";
    $.each(colsSpec,function(row, col) {
        if(row > 0) {    //表头行
            html += ("<tr class=\"tr" + (row%2) + "\">");
            html +=  ("<td width=\"100\"><input type=\"
checkbox\" id=\"col_"+
                col  [Constants.Name] + "\" checked ></
input></td>");
            html += ("<td width=\"100\">"+
                ((typeof  (col  [Constants.Remarks]) == "
undefined")?
                    col   [Constants.Name]:col  [Constants.
Remarks])+"</td>");
                if(col[Constants.isNullable]) {
            html += ("<td><input type=\"text\" id=\"col_"+
            col[Constants.Name]+"_val\"></input></td>");
                } else {
            html += ("<td><input type=\"text\" id=\"col_"+
            col[Constants.Name]+"_val\"></input>*</td>");
                }
                html += ("</tr>");
                //记录各列的名称和类型
                colCount++;
                colNames.push(col[Constants.Name]);
                colTypes.push(col[Constants.Type]);
            }
    });
    html += "</table>";
    //设置 HTML
    $('#content_view').html(html);
});
```

代码 15 中，首先获取表模式信息，再根据表模式信息生成各列的输入控件，最后将 HTML 代码设置给 HTML 容器。

其中为了区分各输入控件，各控件的 ID 与列名进行关联；

此外，为了方便对赋值列的选择，在各输入控件前添加了一复选框，该复选框用于判断该列是否需要设置值。其界面如图 4 所示。

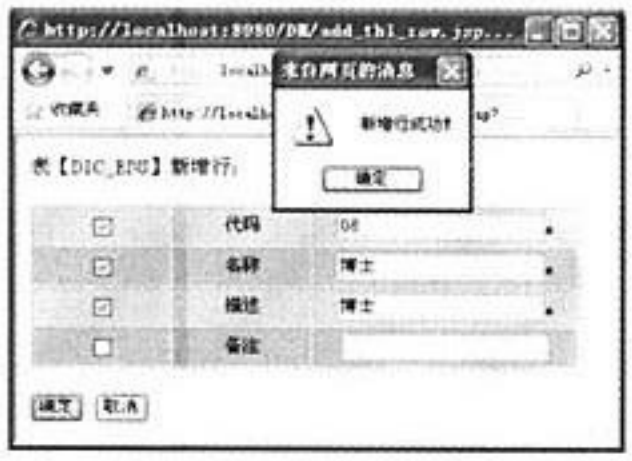


图 4 新增记录行界面

3.3.3 提交新增记录

当提交新增记录时，页面会根据用户选择情况对输入内容进行检查，并对列值进行包装，继而调用新增记录行的服务接口执行数据库操作，并返回结果。其关键处理如代码 16 所示：

代码 16 提交新增记录

文件名:add_tbl_row.jsp

```
$('#btn_ok').click(function() {
    //内容为空检查
    ...
    var cols=new Array();
    var vals=new Array();
    //列筛选
    for(var i = 0; i < colCount; ++i) {
        var chx_id = '#col_'+colNames[i];
        var val_id = '#col_'+colNames[i]+"_val";
        var type = colTypes[i];
        //根据列类型对列值进行包装
        if($(chx_id).attr("checked")=="checked") {
            if ( (type.indexOf ("CHAR")! =-1) || (type.
indexOf("DATE")! =-1) ) {
                vals.push("'"+$(val_id).val()+"'");
            } else {
                vals.push($(val_id).val());
            }
            //列名集合
            cols.push(colNames[i]);
        }
    }
    var params = "tbl=<%=tbl%>&cols="+cols+"&vals="+
vals;
    $.getJSON(encodeURI("/DataServices/addTblRow.ws?"+
params), function(array) {
        alert(getResultDesc("新增行", array["RESULT"]));
        window.history.go(-1);
    });
});
```

代码 16 中，首先根据数据项的选择情况来对空值进行判断，继而获取各列输入的值并按照该列类型对值进行包装，最后根据列名集合和列值集合调用新增记录行的服务接口并反馈执行结果。

4 结语

通过记录行维护和模式信息维护来全面阐述了数据维护平台的实现思路，提出了数据维护服务接口的概念，增强了架构的开发级柔性。此外，该平台结合使用 Ajax 机制和 JSON 格式，对数据维护信息（模式信息和内容信息）进行了集成，实现了自动生成维护界面、平衡了前后端的数据处理压力等功能，实现了用户级的柔性。

需要指出的是，文中提到的部分功能实现并不适用所有数据库（例如：SQLite、SQL Server 等），如何实现对主流数据库的兼容，将是后期亟待解决的课题。

（收稿日期：2012-10-09）

（上接第 40 页）

数据导入成功后，就获得了两个数据表，假设为 table1 和 table2 (注意，数据表的默认名称是根据 Excel 文件名自动命名的，为了后面 SQL 指令编写的方便，可以对其重命名)，两个表格的字段是 SSMS 自动根据 Excel 表首行的列名称建立好的。现在可以方便地使用 SQL 指令来完成数据筛选，例如可以写出以下 SQL 语句：

```
Select * from 表 1 where 字段 3 in  (select 字段 4 from 表 2
where 字段 5>0)
```

这里实际上是嵌套了一个子查询，限于篇幅，具体 SQL 指令的编写方法各位读者可参考其他文献。执行该 SQL 指令后，筛选结果就会出现在 SSMS 的结果窗口，最后还有一个问题，如何将筛选结果以 Excel 表格方式表示，这里有一个简单的方法，可直接利用鼠标在结果窗口中复制出所有查询结果，然后直接粘贴到 Excel 表中，Excel 会自动对其进行分列处理，只需要补充一下标题行就可彻底完成此项筛选任务。

4 结语

通过 SSMS 工具，将多表格 Excel 筛选问题转化为 SQL 查询问题，尤其对那些对 Excel 编程不够熟悉但对 SQL 相对熟悉的人员，不失为一个有效的方案。

（收稿日期：2012-12-18）

AFC 数据库快速备份与恢复策略

刘恒学

摘　要：针对 AFC 数据库后期由于数据的不断增长所导致的备份和恢复时间不断增加的问题，结合数据库热备份及恢复和逻辑备份及恢复各自的优点，提出了解决这一问题的快速备份和恢复的策略，并在 Oracle 数据库中进行实施验证，实践证明效果良好。

关健词：数据库；备份；恢复

一方面，随着时间的推移，轨道交通线路上的客流量不断增加，从而存入 AFC 数据库的数据越来越多，数据库查询速度变得越来越慢，数据备份的时间越来越长，当然恢复的时间也会变得越来越长。虽然通过各种数据库维护措施[1-3]能够对性能有所改善，但不能根本解决这一问题。另一方面，由于数据库系统的硬件也有发生故障的风险，虽说各种技术手段比如磁盘阵列和容灾系统[4]可以减少数据库出现因介质故障导致停机的几率，但采用容灾系统意味着运营成本的增大。因此基于成本的考虑，目前大部分城市的 AFC 系统除 ACC 数据库会建设容灾系统外，线路中心 LC 数据库一般不会建立容灾系统。故对于线路中心数据库来说，不能彻底避免因存储介质故障所导致的数据库不能启动的事件发生。数据库管理员的一块心病是，如何能够做到当数据库发生故障时，在第一时间以最快的速度恢复数据库的正常运行。因此在出现这种事件之前，预先做好应对措施，使得能够在较短的时间内利用数据库的备份进行恢复，使系统能够快速重新投入正常服务，对运营的影响降到最小，是值得探讨的问题。

1　数据库备份与恢复

为了叙述的方便，这里对数据库的备份和恢复做一个简单的介绍。数据库的备份可以分成物理备份和逻辑备份。所谓物理备份是基于存储的物理块的备份。物理备份又分为热备和冷备，前者在数据库正常运行时通过数据库自身提供的工具进行备份，后者在数据库正常关闭退出运行之后使用操作系统的复制命令将数据库相关的文件进行复制所执行的备份。而逻辑备份则是通过数据库提供的工具将数据库运行的某一时刻的逻辑结构和所存数据导出以另一种格式的文件存储。数据库出现故障时，可以通过热备备份的文件恢复到数据库出现故障时刻的一致状态，而通过冷备的文件或者逻辑备份进行恢复则不能恢复到出现故障时刻的一致状态，而只能恢复到备份时刻数据库的状态，并且冷备份只有在数据库未运行的状态下，才能进行恢复，而逻辑备份只能在数据库运行时通过导入工具进行导入。

2　常规备份与恢复策略

一般情况下，为了提高备份和恢复的速度，可以采用下列常规措施：

2.1　优化日常备份脚本

根据业务系统的实际情况通过调整日常备份脚本中命令所涉及的相关参数可以优化提高备份的速度。当然在提高备份速度的同时，特别还要兼顾数据库在使用备份产生的备份集进行恢复时速度是恰当最快的。

2.2　使用快速存储介质存储备份

为了加快备份和恢复的进程，可以使用存储速度较快的存储介质，比如用硬盘代替磁带。例如，对于 Oracle 数据库，为了实现快速的备份与恢复，可以使用闪回恢复区，将备份存储在磁盘而非磁带上，这样恢复的速度相对基于磁带的备份会有较大的提高。

2.3　启用数据块变更跟踪

为了改进增量备份的性能，可以启用数据块变更跟踪功能，这样就不必扫描所有数据文件，只要扫描数据块变更跟踪文件就可以知道哪些数据块需要备份。

2.4　恢复前准确判断恢复的类型

在恢复前，根据数据所出现的数据库错误信息，准确判断数据库发生故障的原因，进而确定恢复的类型，尽量做到能够用较短时间恢复的方法进行恢复，不要发生用较长时间恢复的方案来恢复较短时间就可以恢复的数据库故障。原则上说来，只要热备份没有发生错误，大部分数据库故障都可以通过整个数据库的完全恢复来恢复，可是这时的恢复时间是最长的，因此，在万不得已的情况下，才进行整个数据库的完全恢复。下面列出了各种恢复的优先顺序：

块恢复 > 文件恢复 > 表空间恢复 > 整个数据库恢复

2.5 经常进行各种数据库故障恢复情景的演练

为了减少在生产环境中进行恢复时由于人为判断及操作不

熟练所导致的恢复时间的延误，应经常在测试数据库环境中进行各种数据库故障场景的恢复演练，做到对各种故障的判断以及各种恢复过程烂熟于心。

3 快速备份与恢复策略

虽然通过上述常规手段可以减少备份和恢复的时间，但由于整个数据库发生故障的情况还是有可能发生的，这时就不得不进行整个数据库的完全恢复。因此，针对这种恶劣情形，也要做好必要准备，想方设法减少这种情况下的恢复时间。

一个生产系统最关心的是最近一段时间的数据，而对于已经过去相当长时间的历史数据相对关注度较小，因此有必要将数据分为两类来对待，一类是需要随时在线以备查询或维护的数据，另一类是以备今后查询分析的数据，后一类数据的恢复需求没有前一类数据恢复那么迫切。当然随着时间的推移，原来属于第一类数据中的一部分会变成第二类数据。

一般说来，应用系统的数据除了一部分所谓的大表（比如交易或日志）外，其他表的数据量比较小，一般对于这些大表会采用分区机制进行管理。而应用系统的大部分功能只涉及到小表的数据和大表近期的数据。为此数据库系统的备份恢复策略可以采用结合用逻辑备份恢复和归档模式下的常规热备份（即联机物理备份）与恢复的方式来进行。备份时，使用逻辑备份方式分段备份大表前期的历史数据，而用归档模式下常规热备份方式备份小表的数据和大表近期的数据。在恢复时，先通过归档模式下的常规恢复使得小表的数据和大表近期的数据得以迅速恢复，使得系统可以立即投入运行不至于影响系统的日常业务，在数据库正常运行可以对外提供服务的状态下，逐步将逻辑备份的大表前期的分区数据通过逻辑恢复的方式恢复到数据库中。也就是说，先将第一类数据通过表空间恢复起来，然后将数据库系统置于普通用户的可用状态，对外提供服务，然后再对第二类数据在数据库可用状态下进行恢复，使得普通用户基本感觉不出数据库恢复的漫长过程。

4 配合快速备份与恢复的数据库设计原则

为了配合实施快速备份恢复策略，需要对数据库中的部分表的存储结构进行必要的调整。可以将数据库中的表分成两类，一类为不随时间增长的基础数据表，另一类与业务事件相关的随时间记录不断增多的表。对于前一类表，表的存储结构不需改变采用常规的存储方式，而对于后一类表，则需要对其存储结构进行调整，采用适应记录随时间增长的特殊存储结构，比如分区存储结构。

因为可以针对单独的数据文件或表空间进行恢复，故将单独的一个数据文件或表空间划分为多个尺寸较小的数据文件或表空间，有利于数据库的备份和恢复，因为显然多个文件同时损坏的几率明显低于单独的一个文件损坏的几率。

5 具体实施方案

5.1 备份方案

5.1.1 范围分区

为了将一些历史数据放在只读空间中，需要将那些随着时间的增长记录不断增长的表按日期时间进行范围分区，以便将不再变化的表分区所在的表空间设置成只读状态。

5.1.2 自动维护

可以按照文献[1]的方法编写 PL/SQL 程序包进行数据库表分区的自动维护。

5.1.3 常规工作

为了避免在备份和恢复过程时进行大量的手工操作，故需要编写数据库程序包 alhxws_brready 进行备份恢复方案中常规的工作：比如检查那些不再变化的分区表空间，调用此程序包的 readonlyexpdpts 函数通过执行下列 SQL 语句将其变成只读表空间：

```
alter tablespace tablespace_name read only;
```

然后通过数据导出工具 expdp 将此表空间逻辑备份出来，并且只读表空间只备份一次，必要时可以将导出的文件复制到多个地方保存。

当满足条件的所有表分区所在的表空间变成只读状态并且通过逻辑备份出来之后，调用此程序包的 makerecoversql 函数根据当前的只读表分区利用 Oracle 本身提供的文件处理包 UTL_FILE 自动生成恢复时要用到的脚本 offline_readonly.sql、online_readonly.sql，此函数其中一条关键语句如下：

```
select 'ALTER TABLESPACE ' || TABLESPACE_NAME ||
' DATAFILE OFFLINE;' sqlstatement from dba_tablespaces
where status='READ ONLY' order by TABLESPACE_NAME;
```

当然这个数据库程序包也包括恢复中所要用到的一些例程，比如例程 repairall。

5.1.4 设置任务定时执行

通过 Oracle dbms_scheduler 包创建定期执行的任务，执行上述事先测试通过的数据库程序包中的函数以便将新的只读表空间通过工具 expdp 逻辑备份出来并生成恢复时需要使用的各种脚本。

5.1.5 数据库日常热全备份和增量备份

注意在备份中加入 skip readonly 选项避免对只读表空间的备份，由于不备份只读表空间，而只读表空间占了大部分空间，对于那些近期数据量不是特别大的数据库，可以只做 0 级增量备份（相当于全备份），而不做增量备份。例如，我们公司的 AFC 数据库过去每个星期 1 做 0 级增量备份，而其他天做 1 级累积增量备份，现在采用了新的备份方案后，每天都做 0 级增量备份。这样不管哪天需要数据库介质恢复，只要用当天或前一天的备份做恢复即可，恢复时间自然会较短，并且比较平

稳。不管是哪天出现介质故障，恢复的时间大体在一个稳定的时间区间内。

backup incremental level 0 database skip readonly;

5.2 缺失或损坏的只读表空间处理方法

由于下面的恢复方案中要将不正常只读表空间进行处理使其正常化，本节专门来介绍这种处理方法。

当数据库出现下列类似错误，就说明有数据库文件丢失或损坏：

ORA-01157: 无法标识/锁定数据文件 170 - 请参阅 DBWR 跟踪文件；ORA-01110: 数据文件 170: 'G:\ORACLE\ORADATA\MG\TS_EXCHANGE.DBF'；ORA-01122: 数据库文件 170 验证失败；ORA-01110: 数据文件 170: 'G:\ORACLE\ORADATA\MG\TS_EXCHANGE.DBF'；ORA-01210: 数据文件标头发生介质损坏。

如果一个只读表分区所在的表空间已经损坏或丢失，如果不通过常规的恢复方法要将其变得正常，需要采用下列方法进行处理：

(1) 先将表空间的数据文件下线 (offline)

```
ALTER TABLESPACE  ts_name DATAFILE OFFLINE;
```

(2) 如果数据库未打开时将数据库打开

```
ALTER DATABASE OPEN;
```

(3) 建新的表空间 ts_name_new

(4) 在新的表空间 ts_name_new 建临时表 USERNAME.PTABLE_NAME_EXCHANGE，其表结构除没有分区外与原表 USERNAME.PTABLE_NAME 结构相同，注意此时如果删除表空间 ts_name 会报 ORA-14404 错误。

(5) 将此表与原表空间的表分区交换

```
alter table USERNAME.PTABLE_NAME exchange partition PNAME with table USERNAME.PTABLE_NAME_EXCHANGE;
```

(6) 删除交换到损坏表空间的临时表

```
drop table USERNAME.PTABLE_NAME_EXCHANGE;
```

(7) 将此表空间中的所有对象在新建的表空间 ts_name_new 中重建，然后使得此受损表空间中不存在任何数据库对象，这时可以通过下列语句将此表空间删除，此时再删除表空间 ts_name 会成功，因为其中不包含表分区：

```
DROP TABLESPACE ts_name INCLUDING CONTENTS AND DATAFILES;
```

(8) 然后将新建的表空间换名为原来的受损表空间。

```
ALTER TABLESPACE ts_name_new rename to ts_name;
```

5.3 恢复方案

5.3.1 准备只读表空间数据文件下线脚本和上线脚本

由于不对只读表空间进行热备份，故数据库恢复之后由于可能缺只读表空间或者只读表空间已经损坏，故数据库不能打开，要打开数据库必须先将这些只读表空间的文件下线。由于这种文件可能比较多，因此最好按 5.1.3 中的方法事先生成执行这些操作的脚本。如果平时生成的脚本丢失的话，而数据库的恢复目录数据库运行正常，可以用下列 SQL 脚本生成满足要求的 SQL 脚本 createofflineonline.sql (这里假定恢复目录数据库的用户为 rcmg10)：

```
set heading off;
set feedback off;
spool offline_readonly.sql;
select 'ALTER TABLESPACE ' || TABLESPACE_NAME || ' DATAFILE OFFLINE;'from rcmg10.rc_datafile where read_only=1;
select 'exit;' from dual;
spool off;
spool online_readonly.sql;
select 'ALTER TABLESPACE ' || TABLESPACE_NAME || ' DATAFILE ONLINE;' from rcmg10.rc_datafile where read_only=1;
select 'exit;' from dual;
spool off;
exit;
```

对于下线之后能够上线 (online) 的表空间说明其没损坏，故不必进行处理。

当恢复目录数据库正常时，可执行下列操作系统命令

```
sqlplus -S / as sysdba @createofflineonline.sql
```

生成只读表空间数据文件下线 SQL 脚本 offline_readonly.sql 和上线 SQL 脚本 online_readonly.sql。

5.3.2 用全备份和增量备份集还原和恢复数据库到可用状态

由于全备份和增量备份集的数据量相对较小，故此时数据库文件还原时间很快。

此时要先将自动备份控制文件和 spfile 文件的功能关闭，由于要将大量只读表空间的文件下线或上线，如果不关闭自动备份控制文件功能的话，就会消耗大量无谓的时间，得不偿失。先将 offline_readonly.sql、online_readonly.sql、createimpbatch.sql、createsetreadonlybatch.sql 复制到运行 rman 的目录下，执行下列 RMAN 脚本

```
startup mount;# 启动数据库进入 mount 状态
CONFIGURE CONTROLFILE AUTOBACKUP OFF;# 关闭
# 自动备份控制文件功能,避免浪费大量无谓的时间
restore database;# 还原文件
recover database;# 介质恢复
host "sqlplus -S / as sysdba @offline_readonly.sql" ;# 将
# 只读表空间的文件下线
alter database open;# 打开数据库
 host "sqlplus -S / as sysdba @online_readonly.sql" ;# 将
# 只读表空间的文件上线,此时如果很多只读表空间的文件已经
# 丢失或已经损坏,则这些文件不能上线,提示错误信息,不用管
host "sqlplus -S / as sysdba @createimpbatch.sql" ;# 预
```

```
# 先生成导入数据的脚本
    host "sqlplus –S / as sysdba @createsetreadonlybatch.
sql" ;# 预先生成将表空间设置为只读的脚本
    sql 'declare  i integer; begin i:=alhxws_brready.repairall;
end;';# 将所有损坏的离线表空间进行恰当的处理使其上线,这
# 里 alhxws_brready 为事先实现并经过测试的用于处理相关表
# 空间的 Oracle 包
     CONFIGURE CONTROLFILE AUTOBACKUP ON;# 打开
# 自动备份控制文件功能
```

这里的 SQL 脚本 createimpbatch.sql 内容如下：

```
set heading off;
set feedback off;
SET LINESIZE 255;
spool impreadonlydata.sh;
 select 'impdp system/lhxws DIRECTORY =EXPDIR
DUMPFILE =' || '''' || t1.table_name || substr (t1.
partition_name,2) || '.DMP' || '''' || ' content =DATA_ONLY
LOGFILE =' || t1.table_name || substr (partition_name,2) ||
'_IMP.LOG' || ' STATUS=120 SKIP_UNUSABLE_INDEXES=
YES' || chr(10) || 'sqlplus –S / as sysdba @settsreadonly.sql '
from dba_tab_partitions t1,dba_tablespaces t2 where t1.
table_owner ='SZMTR3AFC' and t1.tablespace_name =t2.
tablespace_name and t2.status='READ ONLY'  order by t1.
partition_name desc,t1.table_name;
spool off;
exit;
```

这里的 SQL 脚本 createsetreadonlybatch.sql 内容如下：

```
set heading off;
set feedback off;
SET LINESIZE 80;
spool settsreadonly.sql;
 select 'alter tablespace ' || t1.tablespace_name || ' read
only;'  from dba_tab_partitions t1,dba_tablespaces t2 where
t1.table_owner ='SZMTR3AFC' and t1.tablespace_name =t2.
tablespace_name and t2.status='READ ONLY'  order by t1.
partition_name desc,t1.table_name;
spool off;
exit;
```

5.3.3 启动各种程序将系统置于对用户提供日常服务的状态

通过上述步骤后数据库已经恢复到可用状态，及时启动各种后台应用程序，将系统置于对用户提供日常服务的状态，此时可以对外宣布业务系统可以使用了。

5.3.4 通过使用逻辑恢复工具 impdp 将 expdp 导出的文件导入到数据库中

在系统对外提供服务的同时，以在线的方式，使用导入工具 impdp 按照时间颠倒的次序，越晚的分区越先导入，直到所有必要分区导入为止。虽然总的完整恢复时间 (如果包括只读表空间的导入时间) 不一定比常规的方案快，但方案使数据库的可用性得到了极大的提高。

因为在第二步已经生成脚本 impreadonlydata.sh，此脚本中有大量类似下列的命令：

```
impdp system/lhxws DIRECTORY = EXPDIR DUMPFILE =
'TD_COMMON_2010_12.DMP'  content  =  DATA_ONLY
STATUS= 120 LOGFILE= TD_COMMON_2010_12_IMP.LOG
SKIP_UNUSABLE_INDEXES=YES
```

执行此脚本，当此脚本执行完成之后再执行 sqlplus –S / as sysdba @settsreadonly.sql，成功执行之后，数据库才真正完全恢复成功。

6 提高恢复只读表分区恢复速度的方法

由于 impdp 的导入速度相对于数据库的热恢复而言，速度相对较慢。因此有必要利用热恢复快速的优点来恢复只读表分区。其方法为，每当将一个表分区设置为只读时，此时在新的表空间中建立一个表结构与分区表相同的表，并且建立结构相同的索引，并将原表分区的数据复制到此新表中，然后将相关的表空间设置成只读，再对这些表空间进行强制热备份，并且将此备份设置成永久状态，然后将这些表空间下线，并将相应的文件删除以便节省空间。到需要恢复时，先将原表分区用前面的方法变成空的可用的状态，然后将对应的表通过热恢复进行还原，然后通过对恢复后的表与表分区进行交换，从而实现较快的只读表分区的恢复。最后将空表下线，并删除相应的文件，回到出现介质恢复前的状态。但这种方法有一个缺点就是数据库系统中存在大量下线的无用表空间。

7 对比实验结果

在 Windowns 平台上对一个数据库文件大小为 124GB (非只读数据文件 36GB 剩下的为只读的数据文件) 的 AFC 数据库 (整个数据库系统安装在移动硬盘上) 进行了常规和快速备份恢复的对比实验，结果如表 1 所示。

表 1

类型	备份	恢复
常规	4 小时 40 分 03 秒	5 小时 49 分 56 秒
快速 (本文)	0 小时 13 分 12 秒 (不备份只读数据文件)	1 小时 46 分 16 秒 (恢复到能对外提供服务为止)
节省	4 小时 26 分 52 秒	4 小时 03 分 40 秒

常规恢复时间=文件还原时间+介质恢复时间

快速恢复时间=非只读文件还原时间+非只读介质恢复时间+特殊处理时间

假定：

t = 还原单位空间的平均时间

(下转第 67 页)

将 Excel 表数据导入 MS SQL Server 数据库表的一种有效方法

魏景东

摘　要：介绍在 B/S 应用系统开发时，将数据从 Excel 表导入到 MS SQL Server2005 数据库表的一种有效方法，结合示例，给出了导入功能的 C# 实现编码。

关键词：VS2008 工具；Excel 软件；MS SQL Server2005 数据库；B/S 架构；C# 语言；数据导入

1　引言

在进行 B/S 应用程序开发时，为保证采集数据的完整与准确，通常采用表单形式让用户填写，作完整性检查后添加到数据库表中。如果采集的数据量很大，项目又很多时，也可通过 Microsoft Excel 进行数据采集，应用程序通常会提供将 Excel 表的数据导入到数据库表中的功能。为确保导入数据完整准确，通常都会对 Excel 中数据的填写格式、必填项目进行规定和说明，并让用户下载预先编制的 Excel 模板，让用户按规定和说明采集数据，在客户端完成必要的校验之后再导入。

下面结合某单位职工情况表“xxzx1.xls”，以“xxzx1.xls”为要导入的 Excel 文件。以 Microsoft SQL Server2005 下 pubs 数据库中的 xxzx1 表为导入目的表，详细说明将 Excel 表数据导入 MS SQL Server 数据库表的一种有效技术实现方法。

为简单起见，假设单位职工情况表在 xxzx1.xls 中的表名为缺省 Sheet1，表头栏为 Sheet1 的第一行，如图 1 所示。pubs 数据库中的 xxzx1 表结构如表 1 所示。

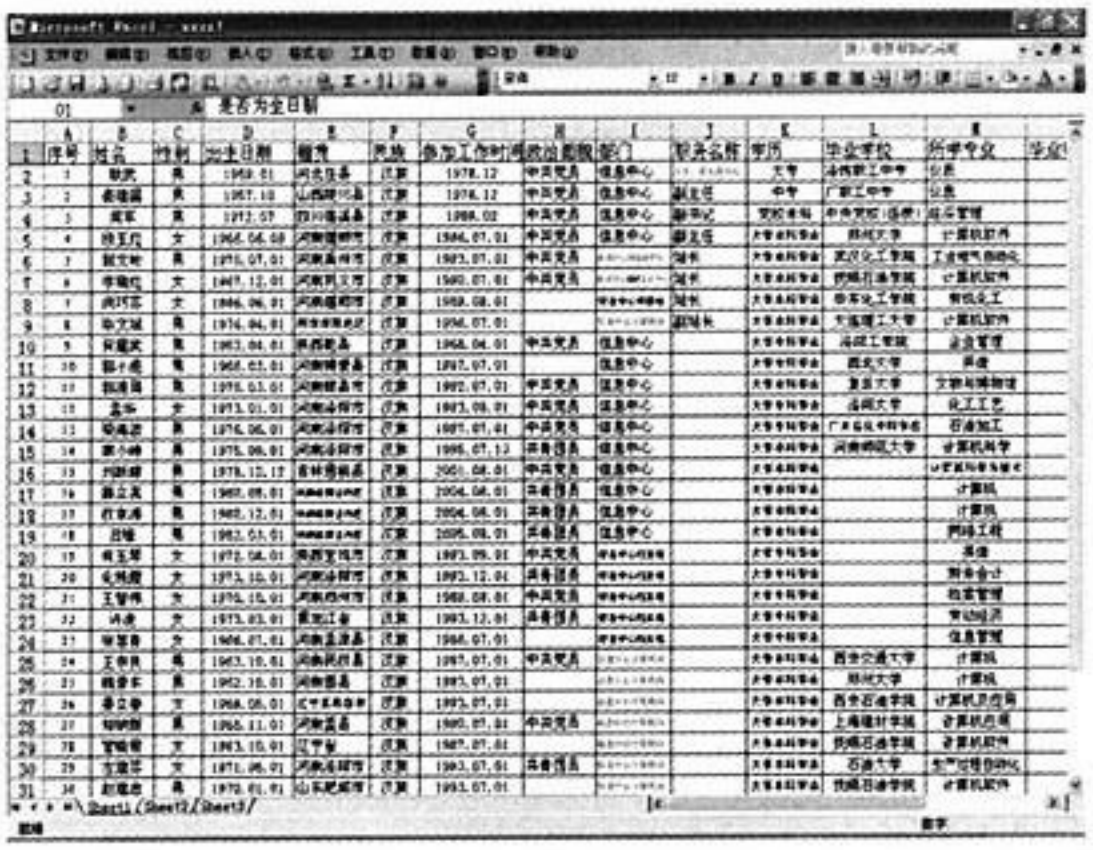


图 1　某单位职工情况表“xxzx1.xls”

表 1　pubs 数据库中的 xxzx1 表结构

列名	数据类型	长度	允许空
序号	nchar	50	主键
姓名	nchar	50	V
性别	nchar	50	V
出生日期	nchar	50	V
籍贯	nchar	50	V
民族	nchar	50	V
参加工作时间	nchar	50	V
政治面貌	nchar	50	V
部门	nchar	50	V
职务名称	nchar	50	V
学历	nchar	50	V
毕业学校	nchar	50	V
所学专业	nchar	50	V
毕业时间	nchar	50	V
是否为全日制	nchar	50	V
备注	nchar	50	V

2 导入 Excel 数据的实现

2.1　创建上传 Excel 文件的页面

启动 Visual Studio 2008，点击“新建”→“网站”→“选择 ASP.NET 网站”→“输入网站位置（语言选 Visual C#）”→点确定创建网站，为简单起见，假设该网站仅创建网页 Default.aspx。

在 Default.aspx 上放置的页面控件及其属性设置如表 2 所示。

表 2 页面控件及其属性设置表

页面控件 ID	Text	onClick
FileUpload1		
Button1	导入数据	Button1_Click
GridView1		

Default.aspx 页面的部局如图 2 所示。

图 2 Default.aspx 页面的设计部局图

其中 FileUpload1 和 Button1 控件用于实现浏览和上传客户端的 Excel 文件，并将 Excel 文的 Sheet1 表内容导入 pubs 数据库中的 xxzx1 表中。GridView1 控件用于展示单位职工情况表“xxzx1.xls”的内容，即导入到 pubs 数据库中的 xxzx1 表中的内容。

2.2 实现编码

打开 Web.config 配置文件，加入下列语据：

```
<appSettings>
<add key = "Conn_link" value = "server =10.123.15.52; database=pubs; uid=sa; pwd=621103"/>
</appSettings>
```

为能够上传、连接、读取 Excel 数据，需在 Default.aspx.cs 的开头加入下列命令：

```
using System.IO;
using System.Data;
using System.Data.OleDb;
using System.Data.SqlClient;
```

在 public partial class _Default : System.Web.UI.Page 段中加入：

```
string strsql_conn = System.Configuration.ConfigurationManager.AppSettings["Conn_link"].ToString();
string file_fullname = "";
```

Excel 数据上传导入功能安排在 Button1_Click 中实现，Excel 数据导入时，根据 Excel 表中的 [序号] 列值，如果 Excel 表中的某行 [序号] 列值与 pubs 数据库中的 xxzx1 表中某行序号字段值相同，就用 Excel 表该行内容更新 pubs 数据库中的 xxzx1 表中序号字段值相同行的内容，并将更新行记数加 1。如果 Excel 表中的某行 [序号] 列值与 pubs 数据库中的 xxzx1 表中任一行序号字段值都不相同，说明该行是新行，就插入到 pubs 数据库中的 xxzx1 表中，并将插入行记数加 1。Excel 数据导入完成后，在 Default.aspx 页面上展示导入的 Excel 数据，并显示更新了多少行，插入了多少新行。Default.aspx 页面的 Excel 数据上传导入功能强大方便。其完整实现编码如下：

```
protected void Button1_Click(object sender, EventArgs e)
    {
        if (FileUpload1.HasFile == false)
        {
            Response.Write("<script>alert('请选择 Excel 文件!')</script>"); return;
        }
        string ext_name = Path.GetExtension (FileUpload1.FileName).ToString().ToUpper();
        if (ext_name ! = ".XLS")
        {
            Response.Write ("<script>alert ('你选择的文件不是 Excel 文件!')</script>"); return;
        }
        string file_name = FileUpload1.FileName;  //获取上传
//文件名
        file_fullname = Server.MapPath("~\\")+file_name;
        FileUpload1.SaveAs(file_fullname);
        string str_Conn = "Provider=Microsoft.Jet.OLEDB.4.0; Data Source = " + file_fullname + ";Extended properties = 'excel 8.0;IMEX=1;HDR=Yes'";
        OleDbConnection excel_Conn = new OleDbConnection (str_Conn);  //IMEX=1 表示将强制混合数据转换为文本；//HDR=YES 表示 Excel 工作表的
        excel_Conn.Open(); //第 1 行为字段名
        DataTable dt = excel_Conn.GetOleDbSchemaTable (OleDbSchemaGuid.Tables, null);
        string[] excelSheets = new String[dt.Rows.Count];
        int k = 0;
        foreach (DataRow row in dt.Rows)
        {
            excelSheets[k] = row["TABLE_NAME"].ToString();
            k++;
        }
        //Response.Write(excelSheets[0]);
        string str_xls = "select * from ["+excelSheets[0]+"]";
//excelSheets[0]的值为第 1 张工作表的名称，
        //excelSheets[1]的值为第 2 张工作表的名称，依次类推
        OleDbDataAdapter ole_adpt = new OleDbDataAdapter (str_xls, excel_Conn);
        DataSet ds = new DataSet();
        ole_adpt.Fill(ds, "[" + excelSheets[0] + "]"); excel_Conn.
```

```
Close();
        int xls_fieldnum = ds.Tables[0].Columns.Count;//电子表
//的列数
            int xls_rows = ds.Tables[0].Rows.Count; //电子表的
//行数
        if(xls_fieldnum==1 || xls_rows==1)
        {
            Response.Write("<script>alert(' 所选 Excel 文件的首
表为空,返回！ ')</script>"); return;
        }
        Response.Write(xls_fieldnum);
        Response.Write("**"); Response.Write(xls_rows);
        GridView1.DataSource = ds.Tables[0];
        GridView1.DataBind();
        SqlConnection conn = new SqlConnection(strsql_conn);
        int k1 = 0;          //新插入记录数
        int j = 0;          //更新记录数
        int i=0;
        while (i < ds.Tables[0].Rows.Count)
        {
        SqlParameter[] param = new SqlParameter[13];
        param [0] = new SqlParameter ("@ 序号", SqlDbType.
NChar);
        param[0].Value =  ds.Tables[0].Rows[i][0];
        param [1] = new SqlParameter ("@ 姓名", SqlDbType.
NChar);
        param[1].Value =  ds.Tables[0].Rows[i][1];
        param [2] = new SqlParameter ("@ 性别", SqlDbType.
NChar);
        param[2].Value =  ds.Tables[0].Rows[i][2];
         param [3] = new SqlParameter ("@ 出生日期",
SqlDbType.NChar);
        param[3].Value =  ds.Tables[0].Rows[i][3];
        param [4] = new SqlParameter ("@ 籍贯", SqlDbType.
NChar);
        param[4].Value =  ds.Tables[0].Rows[i][4];
        param [5] = new SqlParameter ("@ 民族", SqlDbType.
NChar);
        param[5].Value =  ds.Tables[0].Rows[i][5];
         param [6] = new SqlParameter ("@ 参加工作时间",
SqlDbType.NChar);
        param[6].Value =  ds.Tables[0].Rows[i][6];
         param [7] = new SqlParameter ("@ 政治面貌",
SqlDbType.NChar);
        param[7].Value =  ds.Tables[0].Rows[i][7];
        param [8] = new SqlParameter ("@ 部门", SqlDbType.
NChar);
        param[8].Value =  ds.Tables[0].Rows[i][8];
         param [9] = new SqlParameter ("@ 职务名称",
SqlDbType.NChar);
        param[9].Value =  ds.Tables[0].Rows[i][9];
        param[10] = new SqlParameter("@ 学历 2", SqlDbType.
NChar);
        param[10].Value =  ds.Tables[0].Rows[i][10];
         param [11] = new SqlParameter ("@ 毕业学校 2",
SqlDbType.NChar);
        param[11].Value =  ds.Tables[0].Rows[i][11];
         param [12] = new SqlParameter ("@ 所学专业 2",
SqlDbType.NChar);
        param[12].Value =  ds.Tables[0].Rows[i][12];

        SqlParameter[] param1 = new SqlParameter[13];
        param1[0] = new SqlParameter("@ 序号 1", SqlDbType.
NChar);
        param1[0].Value =  ds.Tables[0].Rows[i][0];
        param1[1] = new SqlParameter("@ 姓名 1", SqlDbType.
NChar);
        param1[1].Value =  ds.Tables[0].Rows[i][1];
        param1[2] = new SqlParameter("@ 性别 1", SqlDbType.
NChar);
        param1[2].Value =  ds.Tables[0].Rows[i][2];
         param1 [3] = new SqlParameter ("@ 出生日期 1",
SqlDbType.NChar);
        param1[3].Value =  ds.Tables[0].Rows[i][3];
        param1[4] = new SqlParameter("@ 籍贯 1", SqlDbType.
NChar);
        param1[4].Value =  ds.Tables[0].Rows[i][4];
        param1[5] = new SqlParameter("@ 民族 1", SqlDbType.
NChar);
        param1[5].Value =  ds.Tables[0].Rows[i][5];
        param1 [6] = new SqlParameter ("@ 参加工作时间 1",
SqlDbType.NChar);
        param1[6].Value =  ds.Tables[0].Rows[i][6];
         param1 [7] = new SqlParameter ("@ 政治面貌 1",
SqlDbType.NChar);
        param1[7].Value =  ds.Tables[0].Rows[i][7];
        param1[8] = new SqlParameter("@ 部门 1", SqlDbType.
NChar);
        param1[8].Value =  ds.Tables[0].Rows[i][8];
         param1 [9] = new SqlParameter ("@ 职务名称 1",
SqlDbType.NChar);
        param1[9].Value =  ds.Tables[0].Rows[i][9];
         param1 [10] = new SqlParameter ("@ 学历 21",
SqlDbType.NChar);
        param1[10].Value =  ds.Tables[0].Rows[i][10];
         param1 [11] = new SqlParameter ("@ 毕业学校 21",
SqlDbType.NChar);
        param1[11].Value =  ds.Tables[0].Rows[i][11];
         param1 [12] = new SqlParameter ("@ 所学专业 21",
SqlDbType.NChar);
        param1[12].Value =  ds.Tables[0].Rows[i][12];
         SqlCommand ins = new SqlCommand ("insert into
xxzx1(序号,姓名,性别,出生日期,籍贯,民族,参加工作时间,政治
面貌,部门,职务名称,学历,毕业学校,所学专业) values(@ 序号,@
```

```
姓名,@ 性别,@ 出生日期,@ 籍贯,@ 民族,@ 参加工作时间,@ 政治面貌,@ 部门,@ 职务名称,@ 学历 2,@ 毕业学校 2,@ 所学专业 2)", conn);
        SqlCommand update=new SqlCommand("update xxzx1 set 姓名=@ 姓名 1,性别=@ 性别 1,出生日期=@ 出生日期 1,籍贯=@ 籍贯 1,民族=@ 民族 1,参加工作时间=@ 参加工作时间 1,政治面貌=@ 政治面貌 1,部门=@ 部门 1,职务名称=@ 职务名称 1,学历=@ 学历 21,毕业学校=@ 毕业学校 21,所学专业=@ 所学专业 21 where 序号=@ 序号 1",conn);
            ins.Parameters.Add(param[0]);
            ins.Parameters.Add(param[1]);
            ins.Parameters.Add(param[2]);
            ins.Parameters.Add(param[3]);
            ins.Parameters.Add(param[4]);
            ins.Parameters.Add(param[5]);
            ins.Parameters.Add(param[6]);
            ins.Parameters.Add(param[7]);
            ins.Parameters.Add(param[8]);
            ins.Parameters.Add(param[9]);
            ins.Parameters.Add(param[10]);
            ins.Parameters.Add(param[11]);
            ins.Parameters.Add(param[12]);
            update.Parameters.Add(param1[0]);
            update.Parameters.Add(param1[1]);
            update.Parameters.Add(param1[2]);
            update.Parameters.Add(param1[3]);
            update.Parameters.Add(param1[4]);
            update.Parameters.Add(param1[5]);
            update.Parameters.Add(param1[6]);
            update.Parameters.Add(param1[7]);
            update.Parameters.Add(param1[8]);
            update.Parameters.Add(param1[9]);
            update.Parameters.Add(param1[10]);
            update.Parameters.Add(param1[11]);
            update.Parameters.Add(param1[12]);
            string sqlcheck = "select count(*) from xxzx1 where 序号=@s";//+ds.Tables[0].Rows[i][0].ToString().Trim()+"";
            SqlCommand sqlcmd = new SqlCommand(sqlcheck, conn);
            SqlParameter[] para= new SqlParameter[1];
            para[0] = new SqlParameter("@s", SqlDbType.NChar);
            para[0].Value = ds.Tables[0].Rows[i][0].ToString();
            sqlcmd.Parameters.Add(para[0]);
            conn.Open();
            int count = (int)sqlcmd.ExecuteScalar();
            if (count == 0)
            {
                ins.ExecuteNonQuery(); k1++;
            }
            else { update.ExecuteNonQuery(); j++; }
            conn.Close();
            i++;
        }
        Response.Write("更新记录数:"+j.ToString()+"  ");
        Response.Write("新插入记录数:"+k1.ToString());
    }
```

Default.aspx 页面的运行如图 3 所示。

图 3 Default.aspx 页面运行示意图

3 结语

上述示例在 Windows XP+Windows Server 2003+MS SQL Server 2005 下成功运行，为软件技术人员在 B/S 应用开发中，就如何处理大量 Excel 表数据导入数据库表提供了一种功能强大方便高效的技术实现方法，供软件技术人员在 B/S 应用开发时参考。

参考文献

[1] 电脑编程技巧与维护杂志社. 电脑编程技巧与维护 2010 年合订本. 中国水利水电出版社，2011.

（收稿日期：2013-03-05）

JSP 实现多个关联下拉列表框

李 宇

摘 要：通过一个简单的实例，讲解利用 JSP 进行 Web 开发时多个关联下拉列表的编程实现方法。

关键词：JSP 技术；Web 开发；下拉列表

1 引言

在实际应用中，编写 Web 录入页面时，经常要用到下拉列表框。会有这样的情形，某个下拉列表框的各项与其他的下拉列表框的选择的项相关：

设有 n 个下拉列表框，第一个列表框取数据库中表 test 的字段 field1，第二个列表框取数据库中表 test 的字段 field2，第三个列表框取数据库中表 test 的字段 field3，如此类推，第 n 个列表框取数据库中表 test 的字段 fieldn。第一个列表框取 test 表数据的 select 语句为 select field1 from test；当选择第一个列表框的某一项，如为 FD1，对应的第二个下拉列表框至第 n 个下拉列表框的内容变化，select 语句分别为 select field2 from test where field1 =FD1，select field3 from test where field1=FD1 and field2=第一个 select 语句查询出的第一个结果，……，select fieldn from test where field1=FD1 and field2=第一个 select 语句查询出的第一个结果 and ……and fieldn=第 n-1 个 select 语句查询出的第一个结果；当选择第二个列表框的某一项，如为 FD2，对应的第三个下拉列表框至第 n 个下拉列表框的内容变化，select 语句分别 select field3 from test where field1=FD1 and field2=FD2，……，select fieldn from test where field1 =FD1 and field2=FD2 and field3=第一个 select 语句查询出的第一个结果 and ……and fieldn=第 n-2 个 select 语句查询出的第一个结果；如此类推，当选择第 n-2 个列表框的某一项，如为 FDn-2，对应的第 n-1 个下拉列表框至第 n 个下拉列表框的内容变化，select 语句分别为 select fieldn-1 from test where field1=FD1 and field2=FD2 and……and fieldn-2=FDn-2，select fieldn from test where field1=FD1 and field2=FD2 and……and fieldn-2=FDn-2 and fieldn-1=第一个 select 语句查询出的第一个结果；当选择第 n-1 个列表框的某一项，如为 FDn-1，对应的第 n 个下拉列表框的内容变化，select 语句为 select fieldn from test where field1 =FD1 and field2=FD2 and field3=FD3 …… and fieldn-1=FDn-1。如何在 Jsp 中实现这一功能呢？可用如下方法：

2 实现

建立一个 JSP 文件 prelist.jsp，其关键内容如下：

```
<html>
<head>
<script language=javascript>
function getdata(){
/* 初始化:第一个下拉列表框取第一项,第二个至第 n 个列表框内容刷新 */
 var FD1=document.all.field1.options[0].text;
  document.location.href ="dolist.jsp?FD1 = " +FD1 + "&FD2=&FD3=……&FDn-1=";
}
</script>
</head>
<body onload=getdata()>
<%String st="select field1 from test"
out.println("<select name=field1>")
while (rs.next()){
out.println("<option>"+rs.getString("field1")+"</option>");
}
out.println("</select>");
%>
</body>
</html>
```

建立一个 JSP 文件 dolist.jsp，其关键内容如下：

```
<html>
<head>
<script language=javascript>
Function getdata(f){
/* 选择第一个列表框，则其后的第二个至第 n 个列表框内容都随着其前一个列表框相应变化 */
If (f==1){
  FD1 =document.all.field1.options [document.all.field1.selectedIndex].text;
 document.location.href="dolist.jsp?FD1="+FD1+"&FD2=&FD3=……&FDn-1=";
}
/* 选择第二个列表框，则其后的第三个至第 n 个列表框内
```

```
容都随着其前一个列表框相应变化,第一个列表框保持不变 */
    if (f==2){
      FD1 =document.all.field1.options [document.all.field1.
selectedIndex].text;
      FD2 =document.all.field2.options [document.all.field2.
selectedIndex].text;
      document.location.href = "dolist.jsp?FD1 = " +FD1 + "
&FD2="+FD2+"&FD3=&FD4=……&FDn-1=";
    }
      …
      …
      …
/* 选择第 n-2 个列表框,则其后的第 n-1 个至第 n 个列表框内
容相应变化(第 n 个列表框内容随第 n-1 个列表框变化),第一
个至 n-3 个列表框保持不变 */
if (f==n-2){
FD1=document.all.field1.options[document.all.field1.selectedI
ndex].text;FD2=document.all.field2.options[document.all.fiel
d2.selectedIndex].text
…
FDn-3=document.all.fieldn-3.options[document.all.fieldn-3.
selectedIndex].text; FDn -2 =document.all.fieldn -2.options
[document.all.fieldn-2.selectedIndex].text;
  document.location.href="dolist.jsp?FD1="+FD1+"&FD2="+
FD2+"&FD3="+FD3+……"&FDn-2="+FDn-2+"&FDn-1=";
    }
/* 选择第 n-1 个列表框,则其后的第 n 个列表框内容随第 n-1
个列表框内容相应变化,第一个至 n-2 个列表框保持不变 */
if (f==n-1){
FD1 =document.all.field1.options [document.all.field1.
selectedIndex].text; FD2 =document.all.field2.options
[document.all.field2.selectedIndex].text
…
FDn-2=document.all.fieldn-2.options[document.all.fieldn-2.
selectedIndex].text; FDn -1 =document.all.fieldn -1.options
[document.all.fieldn-1.selectedIndex].text;
      document.location.href = "dolist.jsp?FD1 = " +FD1 + "
&FD2="+FD2+"&FD3="+FD3+……"&FDn-2="+FDn-2+"
&FDn-1="+FDn-1;
    }

}
</script>
</head>
<body>
<%
/* 读取第一个至第 n -1 个各下拉列表框选择的各项 */
String FD1=request.GetParameter("FD1");
…
String FDn-1= request.GetParameter("FDn-1");
/* 初始化变量:从表 test 取各字段值将保存在这些变量中 */
String FLD1="",FLD2="",……FLDn-1="",FLDn="";
/* 建立第一个下拉列表框,可实现当选择某一项时,调用函数
getdata,并传递函数参数 1*/
out.println("<select name=field1 onchange=getdata(1)> ");
/* 从表 test 中取字段 field1 值作为第一个列表框的内容 */
String st="select field1 from test"
while (rs.next()){
  FLD1=rs.getString("field1");
/* 当从表 test 取出的字段 field1 的值等于列表框选择的项时,
加入字段值到第一个列表框中,且第一个列表框显示这一项,否
则只加入字段值到第一个列表框中 */
  if (FD1.equals("FLD1")){
   out.println("<option selected>"+FLD1+"</option>");}
 else{
  out.println("<option>"+FLD1+"</option>");}
  }
/*--------第一个下拉列表框建立完成----------*/
out.println("</select>");
/* 建立第二个下拉列表框,可实现当选择某一项时,调用函数
getdata,并传递函数参数 2*/
out.println("<select name=field2 onchange=getdata(2)> ")
byte counter=0;
/* 从表 test 中取字段 field2 值作为第二个列表框的内容,
select 语句的 where 条件是字段 field1 的值等于第一个列表框
选择的项 */
st="select field2 from test where  field1= ' "+FD1+" ' ";
while (rs.next()){
    FLD2=rs.getString("filed2");
/* 当第二个列表框没有选择的操作,则列表框显示其第一项,且这
第一项的值作为生成第三个列表框的 select 语句的 where 条件
*/
If (FD2.equals("")&&counter==0){
 FD2=FLD2;
 counter++;
       }
/* 当从表 test 取出的字段 field2 的值等于第二个列表框选择的
项时,加入字段值到第二个列表框中,且第二个列表框显示这一
项,否则只加入字段值到第二个列表框中 */
  if (FLD2.equals(FD2){
  out.println("option selected>"+FLD2+"</option>");}
   Else{
 out.println("<option>"+FLD2+"</option>");}
 }
      …
      …
      …
/*---------第二个下拉列表框建立完成----------*/
reponse.println("</select>");
/* 建立第 n-1 个下拉列表框,可实现当选择某一项时,调用函
数 getdata,并传递函数参数 n-1*/
out.println("<select name=fieldn-1 onchange=getdata(n-1)> ")
 counter=0;
/* 从表 test 中取字段 fieldn-1 值作为第 n-1 个列表框的内容,
```

```
select 语句的 where 条件是字段 field1 的值等于第一个列表框选择的项（或第一项），field2 的值等于第二个列表框选择的项（或第一项）……fieldn-2 的值等于第 n-2 个列表框选择的项（或第一项）*/
st="select fieldn-1 from test where field1= ′ "+FD1+" ′ and field2= ′ "+FD2+……+" ′ and fieldn-2= ′ "+FDn-2+" ′ ";
While (rs.next()){
    FLDn-1=rs.getString("filedn-1");
/* 当第 n-1 个列表框没有选择的操作，则列表框显示其第一项，且这第一项的值作为生成第 n 个列表框的 select 语句的 where 条件 */
If (FDn-1.equals("")&&counter==0){
 FDn-1=FLDn-1;
 counter++;
        }
/* 当从表 test 取出的字段 fieldn-1 的值等于第 n-1 个列表框选择的项时，加入字段值到第 n-1 个列表框中，且第 n-1 个列表框显示这一项，否则只加入字段值到第 n-1 个列表框中 */
  if (FLDn-1.equals(FDn-1){
  out.println("option selected>"+FLDn-1+"</option>");}
   Else{
 out.println("<option>"+FLDn-1+"</option>");}
 }
/*---------第 n-1 个下拉列表框建立完成---------*/
out.println("</select>");
/* 建立第 n 个下拉列表框 */
out.println("<select name=fieldn>");
/* 从表 test 中取字段 fieldn 值作为第 n 个列表框的内容，select 语句的 where 条件是字段 field1 的值等于第一个列表框选择的项（或第一项），field2 的值等于第二个列表框选择的项（或第一项）……fieldn-1 的值等于第 n-1 个列表框选择的项（或第一项，由语句
If (FDn-1.equals("")&&counter==0){
 FDn-1=FLDn-1;counter++;}生成）*/
st="select fieldn from test where field1=′ "+FD1+" ′ and field2= ′ "+FD2+……+" ′ and fieldn-1= ′ "+FDn-1+" ′ ";
While (rs.next()){
     FLDn=rs.getString("filedn");
 out.println("<option>"+FLDn+"</option>");
 }
out.println("</select>");
/*---------第 n 个下拉列表框建立完成----------
*/
%>
</body>
</html>
```

具体以 3 个下拉列表框为例说明。

3 具体实例

假设数据库中有个表 area，有字段 province、city、district，表中内容如表 1 所示。

表 1 area 表内容

province	city	district
广东	广州	海珠
广东	广州	越秀
广东	广州	白云
广东	佛山	南海
广东	佛山	顺德
广东	佛山	三水
湖南	长沙	芙蓉
湖南	长沙	天心
湖南	长沙	雨花

建立一个 JSP 文件 prelist.jsp，内容如下：

```
<html>
<head>
<script language=javascript>
function getdata(){
 var province1=document.all.province.options[0].text;
          document.location.href="dolist.jsp? province1="
+ province1+"&city1=";
}
</script>
</head>
<body onload=getdata()>
<%
String st="select province from area";
out.println("<select name=province>");
while (rs.next()){
out.println("<option>"+rs.getString("province")+"</option>"
}
out.println("</select>");
```

建立一个 jsp 文件 dolist.jsp，其关键内容如下：

```
<html>
<head>
<script language=javascript>
function getdata(f){
if (f==1){
province1=
document.all.province.options[document.all.province.selecte
dIndex].text;
document.location.href="dolist.jsp? province1="+ province1+"
&city1=";
    }
if (f==2){
province1=
document.all.province.options[document.all.province.selected
Index].text;
    city1=document.all.city.options[document.all.city.selecte
```

```
dIndex].text;
                document.location.href="dolist.jsp? province1="
+ province1+"&city1="+city1;
    }
    }
    </script>
    </head>
    <body >
  <%
/* 为避免出现中文乱码，将取出的字符串进行代码转换 (iso-
8859-1 转为 gbk)*/
String province1= new String
(request.getParameter ("province1").getBytes ("iso -8859 -1"),"
gbk");
     String city1= new String
(request.getParameter("city1") .getBytes("iso-8859-1"),"gbk");
     String province="",city="",district="";
out.println("<select name=province onchage=getdata(1) >");
     String st="select province from area";
     while(rs.next()){
     province=rs.getString("province");
     if (province1.equals(province)){
      out.println("<option selected>"+province+"</option>");}
     else{
     out.println("<option>"+province+"</option>");}
     }
      out.println("</select>");
   out.println("<select name=city onchage=getdata(2) >");
       byte counter=0;
st="select city from area where province=' "+province1+" ' ";
     while(rs.next()){
     city=rs.getString("city");
     if (city1.equals("") && counter==0){
     city1=city;
     counter++;
       }
     if (city1.equals(city)){
      out.println("<option selected>"+city+"</option>");}
     else{
     out.println("<option>"+city+"</option>");}
     }
     out.println("</select>");
     out.println("<select name=distrinct >");
      st = "select distrinct from area where province =' " +
province1+" ' and city=' "+city1+" ' ";
     while(rs.next()){
      out.println("<option>"+ district +"</option>");
     }
     out.println("</select>");
     %>
     </body>
     </html>
```

运行 prelist.jsp，得到如下结果：

[广东] [广州] [海珠]

若选择第二个列表框为“佛山”，则结果如下：

[广东] [佛山] [南海]

说明：以上所例举的表 test 和 area 只是为了说明 Jsp 如何实现相互关联下拉列表框的方法，实际的数据库表设计这样设计是不合适的。如表 area 应分为 3 个表，设为 province，city 和 distrinct。表 province 有两个字段 pID 和 rovinces,city 有 3 个字段 pID、cID 和 cities，表 distrinct 有 3 个字段 pID，cID 和 distrincts。3 个表内容如图 1 所示。

province

pID	provinces
1	广东
2	湖南

city

pID	cID	cities
1	1	广州
1	2	佛山
2	1	长沙

district

pID	cID	districts
1	1	海珠
1	1	越秀
1	1	白云
1	2	顺德
1	2	南海
1	2	三水
2	1	芙蓉
2	1	天心
2	1	雨花

图 1 表关系

建立第一个下拉列表框的 select 语句为" select pID, provinces from province"；假设查询出的字段 pID 的值保存在变量 pIDlist 中，字段 provinces 的值保存在变量 provinces 中，则建立第一个下拉列表框的关键代码为 out.println (" <option>" + provinces+" </option>")，建立第二个下拉列表框的 select 语句为" select pID,cID,cities from province a,city b where a.pID=b.pID and a.pID=' " +pIDlist+" ' "；假设查询出的字段 pID 的值保存在变量 pIDlist 中，字段 cID 的值保存在变量 cIDlist 中，字段 cities 的值保存在变量 cities 中，则建立第二个下拉列表框的关键代码为 out.println (" <option>" +cities+" </option>")，建立第三个下拉列表框的 select 语句为" select distrincts from province a,city b, distrinct c where a.pID=b.pID and b.pID=c.pID and b.cID=c.cID and a.pID=' " +pIDlist+" ' and b.cID=' " +cIDlist+" ' "；假设字段 distincts 的值保存在变量 distrincts 中，则建立第三个下拉框的关键代码为 out.println (" <option>" +distrincts+" </option>")。

4 结语

通常还会遇到简单 select 查询语句不能生成下拉列表框的各项，需要编写存储过程，这时就用调用存储过程的语句代替简单的 select 查询语句，而程序中运用的关键方法无需改变，这里不再赘述了。

（收稿日期：2012-11-29）

使用 FreeTextBox 和 ASPJpeg 增强网页图文管理

卢增云

摘　要：从实际应用出发，简要介绍了目前常见的网页管理系统的存储特点，并针对早期网页管理系统所存在的不足，提出了一种基于 ASP.NET 动态网页技术的图文混编文章存储技术。该技术主要利用 ASP.NET 开源服务器控件 FreeTextBox 和 ASPJpeg 实现文章的在线编辑。

关键词：网页图文管理；FreeTextBox 控件；ASPJpeg 控件

1　引言

网页图文管理系统通常扮演着互联网上信息发布的后台维护的角色，合法用户通过 IE 浏览器可登录系统发布和管理网页文章；早期的使 ASP 开发的网站的后台维护系统不是采用富文本的在线编辑方式，而是采用纯文本的编辑方式，图片和附件另外上传到指定的目录，前台展现的时候通过读取数据库的文本内容和相关的图片和附件的内容显示出来，可以看到这种方式表现出来的网页比较单调，文字的字体单一，图片的排版方式不能灵活变化。这类网站需要更新后台网页维护和前台的展示方式，后台的文章管理需要使用富文本的编辑方式来实现，前台展示方式需要从原来的 table 方式更新到 DIV+CSS 方式。

因此，需要一种新的网站的后台维护方式和存储技术来弥补以上的不足，同时增强网站的安全性。这种技术能实现图文混编并把文章保存到数据库中。这就是要介绍的利用 ASP.NET 控件 FreeTextBox 和 ASPJpeg 实现网页图文的在线编辑和存储技术。

2　关键技术

ASP.NET 是一种建立动态 Web 应用程序的技术。与其他网络语言相比，ASP.NET 具有方便、灵活、生产效率高，安全性高及构件完整性强等特性，ASP.NET 程序中允许使用 Javascript 来操作网页之间的交互。利用 SQL Server 2008 数据库和 Asp.net 动态网页技术来保存网页文章信息，可以很大程度上减少网站管理人员的工作量，提高工作效率和系统的安全性。FreeTextBox 和 ASPJpeg 的结合可以完美实现 HTML 网页图文的编辑管理。保存网页文章的时候，文本和图片分开保存，编辑器中内容自动转换成 html 代码并保存到数据库中，而图片则存入对应的文件夹。

2.1　数据结构设计

设计两个表 inform 和 informAnnex（如表 1 和表 2 所示）用于记录网页内容和每条网页内容与图片 ID 的关联，一条 inform 中的每个图片都记录在 informAnnex 表中，通过 InformID 字关联，因为在 informAnnex 表中的 ID 是唯一的，所以上传的图片可以方便地使用不重复的 ID 来命名，如果是小的缩略图就在 ID 后面加上“_l”标识。

表 1　inform 数据结构

字段	类型	描述
ID	int	唯一标识，设为主键
Kind	int	分类 ID
UserID	int	用户 ID
Title	varchar	题目
Content	ntext	内容
CreateDate	datetime	创建时间

表 2　informAnnex 数据结构

字段	类型	描述
ID	int	唯一标识，设为主键
InformID	int	对应的文章的 ID
Name	varchar	原始文件名
FileSize	bigint	文件大小
CreateDate	datetime	上传时间
ImgPath	varchar	文件路径

2.2　FreeTextBox 及图片上传控件的使用

ASP.NET 本身没有提供富文本编辑器控件，要做一个网页后台管理软件就需要使用第三方的富文本控件，这里选择了目前使用较多的 FreeTextBox，下载地址：http://freetextbox.com，堪称第一款开源免费.Net 编辑器，在使用上比较简单，综合了 Word 界面，拥有较全的功能以及多语言支持而得到广大开发人员的青睐。FreeTextBox 使用 3.3 版本，命名空间 FreeTextBoxControls。

经过多方设置比较，发现控件内置的图像上传控件使用不

方便，该控件仅简单实现了上传功能，如果再次上传的图片文件名与已存在的文件名同名的话，空间上已有的图片将会被覆盖；图片的保存位置不能方便设置，上传的图片大小不能压缩；空间图片多了，预览图很慢，查找不方便；上传的附件类型不能设置，带来一些安全隐患等；所以决定重新编写 ftb.imagegallery.aspx 用于图片上传，从而避免原来图片上传控件的缺点。

编写 ftb.imagegallery.aspx 过程中，发现了几个难点：打开 ftb.imagegallery.aspx，在这个图像上传的页面中选择图片后点击“插入”，原来页面的编辑窗口不能接收插入的图像，原因在于原来的 FTB 的没办法关联 ftb.imagegallery.aspx 中的控件，无法使用 FTB_API，所以不能使用内置的 ToolBar 中的“插入图片”的按钮，决定采用自定义的工具栏，在引用<FTB:FreeTextBox …… >处声明 AutoGenerateToolbarsFromString = "false"，点击“插入图片”按钮后打开 ftb.imagegallery.aspx，自定义的工具栏的按钮的语句：

<FTB:ToolbarButton ScriptBlock = " var winValue = window.showModalDialog ('ftb.imagegallery.aspx? ftb=FreeTextBox1', null, 'dialogHeight =550px,dialogWidth =550px,center =1') ;if (winValue.indexOf ('.') >0) {this.ftb.InsertHtml (winValue) ;}" ButtonImage=" insertimage" runat=" server" />ScriptBlock

等号后面的是 JavaScript 语句，父窗口等待由 showModalDialog 方法打开的子窗口返回 winValue 值，关闭 ftb.imagegallery.aspx 窗口后将 winValue 文本插入到父窗口的光标位置的 HTML，这样图像插入到 FreeTextBox 窗口。注意这里只能使用 showModalDialog 方法，不然 ftb.imagegallery.aspx 的返回值不能被 FreeTextBox 接收。

对于 FreeTextBox 的上传图片和附件的操作都在 ftb.imagegallery.aspx 中完成，其中用到的关键技术是：利用返回的 returnValue 将图片或附件插入到相应的位置。在 ftb.imagegallery.aspx 返回语句的写法：

```
function returnImage(imagename,width,height) {
      var arr = new Array();
var img = "<P align=center><IMG border=0 alt=" + imagename
+" src="+ imagename + "  width="+ width  +"></p>";
      window.parent.returnValue = img;
      window.parent.close();
}
```

图像保存的路径的设置办法，使用系统时间的年月日和模块 ID 结合的方式，这样可以保证路径不重复而且容易查找。模块 ID 这两个页面之间的传值通过 Session [" ModuleID"] 来实现。

```
public string GetPath()
{
      string path = "FTBimages/"; //事先创建的虚拟目录
      DateTime now = DateTime.Now; // 定义时间变量并赋
//值为当前时间
      string year = now.Year.ToString().Trim();// 获取当前时间
//的年份,下同
      string month = now.Month.ToString().Trim();
      string day = now.Day.ToString().Trim();
      string hour = now.Hour.ToString().Trim();
      string filepath = path + year + "_" + month + "/" + day+
"/" + Session["ModuleID"];
      // 获取虚拟路径对应的物理路径
if (! System.IO.Directory.Exists(Server.MapPath(filepath)))
                System.IO.Directory.CreateDirectory (Server.
MapPath(filepath));
       Session["ImgPath"]= filepath; //用于保存图片路径,提供
//给 ftb.imagegallery.aspx 使用
      return filepath;
}
```

允许上传附件类型的设置办法：

```
private string[] AcceptedFileTypes = new string[] {"jpg","jpeg","
jpe","gif","png"};
private bool IsValidFileType(string FileName) {
string ext = FileName.Substring(FileName.LastIndexOf(".")+1,
FileName.Length-FileName.LastIndexOf(".")-1);
for (int i=0; i<AcceptedFileTypes.Length; i++) {
if (ext.ToLower() == AcceptedFileTypes[i]) {
  return true;
}
}
      return false;
}
```

2.3 ASPJpeg 的使用

网站维护中上传的图片需要生成缩略图和加入网站版权水印。AspJpeg 是一款功能相当强大的基于 Microsoft IIS 环境的图片处理组件。下载地址：http://www.aspjpeg.com，AspJpeg 可以使用很少的代码在 ASP/ASP.Net 应用程序上动态地创建高质量的缩略图像，支持的图像格式有：JPEG, GIF, BMP, TIFF, PNG。AspJpeg 可以轻松地做到：生成缩略图片、生成水印图片、图片合并。

使用图像管理控件 ASPJpeg 之前先把 ASPJPEGLib.dll 拷贝到网站目录 \bin。这样在使用 UploadFile 控件上传了图片之后可以马上使用 ASPJpeg 对原图压缩、形成缩略图、加入网站版权的水印。具体 asp.net 的语句为：

```
      …
     //保存原图
      UploadFile.PostedFile.SaveAs  (UploadFileDestination +
UploadFileName);
      //声明 ASPJpg 对象
      ASPJPEGLib.IASPJpeg objJpeg;
      objJpeg = new ASPJPEGLib.ASPJpeg();
      // 打开原图
```

```
objJpeg.Open( UploadFileDestination + UploadFileName);
    //计算压缩后的大小
    int w,h;
    float k;
    w=objJpeg.OriginalWidth, h=objJpeg.OriginalHeight;
    k=(float)h/w;
    if (w>1024){
        objJpeg.Width = 1024;
        objJpeg.Height =(int) (k*1024) ;
    }
    objJpeg.Canvas.Font.Color = 0xFF0000; // red
    objJpeg.Canvas.Font.Family = "Courier New";
    objJpeg.Canvas.Font.Bold = 1;
    objJpeg.Canvas.Font.Quality = 4; // Antialiased
    objJpeg.Canvas.Print ( objJpeg.Width -160, objJpeg.
Height-43, "www.fjjty.cn", Missing.Value );
    //重新命名保存
    objJpeg.Save (UploadFileDestination + ID + "." +
GetFileExtends(UploadFileName));
    //这里形成一个缩略图
    ...
```

2.4 FreeTextBox 的安全性问题

在 Web 应用的开发过程中，安全性的意识贯穿始终。在 FreeTextBox 的使用中，最不能忽略的是应该是该控件的安全漏洞问题。发现控件自带的图片浏览上传网页 ftb.imagegallery.aspx 没有设置访问权限，在百度中搜索了下 inurl:ftb.imagegallery.aspx，一打图片肉鸡出现在眼前，虽然 ftb.imagegallery.aspx 在客户端与服务端设有上传文件格式验证，只能上传图片，但任何人访问这个地址就能做上传删除等操作，未免太荒谬了，所以必须在 Page_Load 下设置用户登入验证，方法如下：

编辑 ftb.imagegallery.aspx，在程序代码：

```
 private void Page_Load (object sender, System.
EventArgs e) {
    string isframe = "" + Request["frame"];
        if (isframe ! = "" && ! string.IsNullOrEmpty(Session
["ImgPath"].ToString())) {
RootImagesFolder.Value = Session["ImgPath"].ToString();
CurrentImagesFolder.Value=Session["ImgPath"].ToString();
  ...
```

下面添加你站点的验证，验证不通过直接跳回首页或登入页面，这个验证代码各站点不同，需自己编写，一般都是 session、cookies 验证或读取 SQL 数据库。

同时，把当前图片目录和图片根目录都设置到允许操作的图片目录，这样还可以防止目录越级查看、操作图片和附件文件。

3 实际应用

通过以上介绍的技术，网站的后台维护来的方便灵活了，原来 Word 的文章的内容可以通过“复制”“粘贴”操作，把除图片以外的内容添加到编辑器中，保持格式不变。每一个网页只要浏览的速度允许，都可以插入多个图片等信息，插入后图文可以自由混编，任何文本格式的文章均不需进行复杂的格式转换，便可用统一的 HTML 格式保存到数据库中，运行效果如图 1 和图 2 所示。

图 1 插入图片目录的缩略图

图 2 插入图片后的图文略图

4 结语

选择 Asp.net 控件制作富文本网页编辑器时，除了使用方便合理之外，更重要的要考察控件的安全性；在采用富文本编辑器控件 FreeTextBox 制作网页图文后台管理的过程中，接触了一些控件如 CuteEditor、RadEditor、FCKEditor 也可以方便地实现网页富文本编辑，各种控件的用法和特性还需进一步的对比和验证。随着 Web 应用的增加，介绍的方法还可以扩展，如在 FreeTextBox 中增加按钮，插入 flash、视频等。

参考文献

[1] 林碧英，陈达峰. 自定义分章图文混编文章存储技术的研究. 计算机系统应用，2009，(4)：158.

[2] 房大伟，吕双，刘云峰. ASP.NET 编程宝典（C#）. 人民邮电出版社，2011.

[3] http://www.cnblogs.com/liuzemin/archive/2012/03/31/2426754.html.

（收稿日期：2012-11-07）

MongoDB 和 Node.js 的邮件收发系统

李臣龙 戴汶倖

摘 要：邮件收发系统的功能主要分为用户和管理员两大部分。其中，用户的功能主要包括用户个人信息的管理、邮箱帐户的管理、邮件的接收以及发送等；管理员的功能主要包括管理用户信息、删除用户、管理后台系统等。

关键词：电子邮件；邮件服务；Web Mail 邮件；POP3 协议；SMTP 协议

随着互联网的进一步发展，电子邮件已经成为人们联系沟通的重要手段。传统的 C/S 结构的电子邮件桌面客户端软件（如 Outlook 和 Foxmail 等），在使用前用户需要进行一系列复杂的设置，且不能跨平台使用。基于 B/S 结构的 Web 邮件服务，不仅为用户省去了维护与升级客户端软件的麻烦，更重要的是可以在各种操作系统甚至是移动终端中使用。只要可以接入互联网就可以通过浏览器来收发并管理邮件，提高了用户体验。

本邮件收发系统在实现基本的对电子邮件的接收和发送等功能的基础上，还支持单个用户添加多个邮箱帐户并统一进行管理等功能。用户添加了多个邮箱帐户后可以分别接收各个邮箱中的邮件，并且在发邮件的时候可以选择使用任一邮箱发送。功能测试和验证结果表明，该系统稳定、可扩展，达到实时性要求。

1 技术简介

本邮件收发系统通过 POP3 协议和 SMTP 协议实现对电子邮件的接收和发送，按照 RFC 822 协议所定义的格式对邮件实体进行解析。通过前期的技术选型，最终确定整个系统采用B/S架构，服务器端程序采用 Node.js 作为主要的开发语言，数据库系统采用 MongoDB。

Node.js 是一个 JavaScript 的运行环境，采用 C++编写而成。之所以选择 Node.js，主要是因为通过前期调研了解到邮件收发系统是一个 I/O 密集型 (I/O bound) 的应用。而 Node.js 的无阻塞 (Non-Blocking) I/O、事件驱动 (Event-Driven) 等特性恰好适用于该种应用场景。且 Node.js 采用 Google 的 V8 JavaScript 引擎，确保了其拥有良好的性能。

MongoDB 是一个基于分布式文件存储的数据库。由 C++语言编写。其特点是高性能、易部署、易使用，存储数据非常方便。本系统采用 MongoDB 的主要原因是 MongoDB 原生支持 Node.js，可以提升开发效率。

2 收发原理

如图 1 所示，在电子邮件传递的过程中，发件者的邮件代理从本地通过 SMTP 协议将邮件发送到 SMTP 邮件服务器，邮件服务器通过 SMTP 协议将邮件发送到收件者的 POP3 邮件服务器。收件者的邮件代理使用 POP3 协议将邮件从其 POP3 邮件服务器拉取到本地。

而本系统在整个过程中则起到邮件用户代理这一角色，本系统所部属的服务器本身不具备邮件发送功能，需要用户将自己的邮箱帐户配置到系统中，由本系统来代为发送和接收。

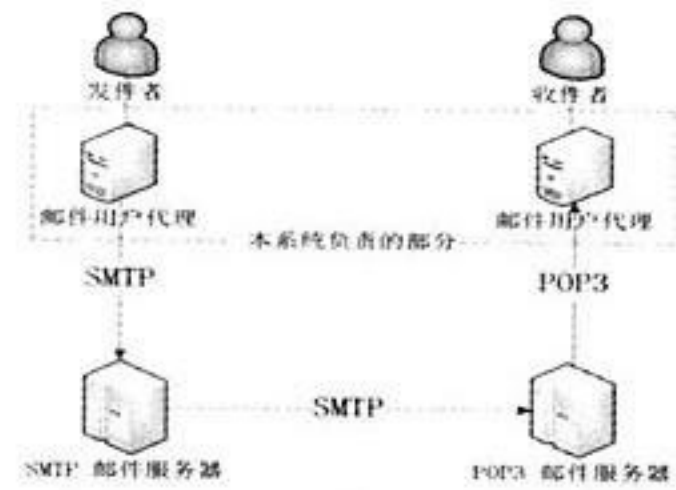


图 1 邮件收发原理

3 需求分析与功能设计

邮件收发系统的应用功能应包括以下几个方面：用户基本信息管理，用户邮箱帐户管理，邮件的接收、发送，草稿邮件管理和联系人管理。

根据需求分析可以设计出系统的功能模块，各个系统功能模块之间的关系如图 2 所示。

图 2 系统功能模块

3.1 用户登录和注册

(1) 根据用户所属的不同角色显示不同的操作页面。管理员登录时，除了显示普通用户所展示的界面之外，还显示管理相关的入口链接。

(2) 新用户注册后成为普通用户。管理员在管理界面可以将普通用户提权为管理员。超级管理员有最高级权限，且自身不可被其他管理员操作，由系统配置文件定义。

3.2 用户角色管理

角色管理主要是给注册的用户分配角色，即修改用户的角色，删除用户的角色。

3.3 用户管理

添加用户，修改用户，删除用户。

3.4 邮件收发及管理

(1) 邮件接收，对用户各个邮箱帐户中新邮件的接收，统一查看用户已下载的所有邮件，接收邮件中的附件等。未阅读的新邮件会高亮显示。

(2) 邮件发送，用户可以指定一个自己的邮箱帐户来发送邮件，在邮件中可以添加附件。邮件发送后将被保存在“已发送”信箱中以备查。

(3) 邮件管理。可以删除或者永久删除邮件，已删除的邮件将移入“已删除”目录以便恢复，永久删除的邮件不能恢复。

3.5 邮箱帐户管理

用户可以添加、编辑或删除邮箱帐户，用户可以使用已添加的邮箱帐户收发邮件。

3.6 联系人管理

(1) 用户可以在联系人页面中添加、编辑、删除联系人信息。

(2) 用户在发送邮件的时候可以直接从联系人列表中快速选取收件地址。

4 实现与测试

本系统最核心的部分就是实现对邮件的接收和发送。在这个过程中主要使用 POP3 协议和 SMTP 协议来达到对电子邮件的接收和发送，并且通过实现 RFC822 协议来解析邮件实体。

用户在使用本系统之前需要添加自己的邮箱帐户，包括输入自己邮箱的用户名、密码和邮件供应商提供的邮件服务器(POP3 和 SMTP 服务器) 地址以及对应端口。之后在每次的收件和发件过程中，系统将和邮件服务器建立起一个或多个 socket 连接，并且按照 POP3 和 SMTP 协议所规定的数据格式，将相应指令或数据通过该 Socket 连接发送到邮件服务器。而邮件服务器则会按照协议规定，返回相应的指令和数据。系统在收到服务器的响应后，即开始按照 RFC 822 协议中的规定，将收到的数据包进行解析或者是进行相应的错误处理。

以下是系统核心部分的代码，实现系统收发核心功能。

```
exports.pull = function (req, res) {
  var uid = req.session.user;
  var resp = {'stat': statCode['UNKNOWN']};
  var box = req.params.mailbox;
  accountModel.findOne(uid, box, function (err, account) {
    var opt = {};
    opt.username = account.email;
    opt.password = account.password;
     opt.port = account.pop_port === undefined ? 110 :
account.pop_port;
    opt.host = account.pop_svr;
    opt.tls = account.pop_ssl || false;
    var options = {
      ignoretlserrs: false,
      enabletls: (opt.tls === true ? true : false),
      debug: (config.debug === true ? true : false),
    };
    var client = new POP3Client(opt.port, opt.host, options);
    // 列出邮件标识
    client.on ('list', function  (status, msgcount, msgnumber,
data, rawdata) {
      if (status === false) {
        client.quit();
        resp.stat = statCode['REMOTE_CONN'];
        return res.json(resp);
      }
      if (msgcount === 0) {
        resp.stat = statCode['NO_MAIL'];
        return res.json(resp);
      }
      sizes = data;
      client.uidl();
    });
    var ep = new EventProxy();
    var events = ["uniqs", "tokens"];
    ep.all(events, function (uidls, tokens) {
     //将邮件标识符与数据库中记录做匹配,判断是否为新邮件
      freshes = [];
      for (var i = 1, l = uidls.length; i < l; i++) {
        if ((! sizes[i] || sizes[i].length <= 6) && ! tokens[uidls[i]])
{
          freshes.push(i);
        }
      }
      if (! freshes.length) { // 没有新邮件则直接返回
        resp['stat'] = statCode['SUCCEED'];
        return res.json(resp);
      }
      uniqs = uidls; // 有新邮件则获取完整的新邮件并保存
```

```
    index = freshes.length - 1;
    client.retr(freshes[index]);
  });
    client.on ("uidl", function (status, msgnumber, data,
rawdata) { // 获取服务器端邮件的唯一标识符
    if (! data || ! data[1]) {
      return res.json(resp);
    }
    ep.emit("uiqs", data);
  });
    mailModel.findTokens(uid, box, function (err, tokens) {
// 获取数据库中邮件的唯一标识符
    if (err) {
      console.log(err);
      return res.json(resp);
    }
    var data = {};
    for (var i = 0, l = tokens.length; i < l; i++) {
      data[tokens[i].token] = true;
    }
    ep.emit("tokens", data);
  });
    client.on ('retr', function (status, msgnumber, data,
rawdata) {// 返回一封完整的邮件
    if (status === false) {
      client.quit();
      return res.json(resp);
    }
    var mpOptions = {defaultCharset: 'UTF-8',
streamAttachments: true};
    var mailparser = new MailParser(mpOptions);
    mailparser.write(data);
    mailparser.end();
      mailparser.on('attachment', function (attachment) {
// 解析到附件时触发事件
      parseReceivedAtta(req.session.user, attachment);
    });
      mailparser.on('end', function (parsed) { // 邮件解析结
// 束后触发事件
      var o = getParsedMail(req.session.user, parsed);
      // 将新邮件存入数据库
      var data = {'uid': uid, 'box': box, 'token': uniqs[freshes
[index]], 'stat': mailStat ['AVAILABLE'], 'new': true, 'content':
o, 'date': new Date(parsed.headers.date)};
      mailModel.save(data, function (err, result) {
        var text = data.content.text ? data.content.text : data.
content.html ? data.content.html : '';
        data.content.des = text.toSafeText();
        data.date = data.date.format('yyyy-MM-dd');
        mails.push(data);
        // 检查是否最后一封新邮件,如果不是则继续处理,否则
// 向前端输出
        if (--index >= 0) {
          return client.retr(freshes[index]);
        }      resp.stat = statCode['SUCCEED'];
        resp.mails = mails;
        return res.json(resp);
      });…
```

在本系统的开发过程中，主要使用单元测试来确保代码中各个对象和函数对各种输入数据所得到的输出的正确性。

在计算机编程中，单元测试是针对程序模块（软件设计的最小单位）来进行正确性检验的测试工作。程序单元是应用的最小可测试部件。在过程化编程中，一个单元就是单个程序、函数、过程等；对于面向对象编程，最小单元就是方法，包括基类（超类）、抽象类、或者派生类（子类）中的方法。

本系统的单元测试使用 Node.js 的第三方单元测试模块 Mocha 完成。在此过程中第三方断言模块使用 Should.js。代码覆盖率报告使用 jscoverage 生成。

5 系统界面设计

进入系统后，可看到本系统的主要用户界面。在界面的上方，是系统相关的导航条。用户可以进入管理账户页面和设置用户个人信息的页面，如果是管理员用户还可看到进入管理的入口。界面右上角是搜索邮件的搜索框，用户可以键入关键字并搜索自己收件箱中的邮件。

在界面的左侧，是邮箱相关功能的工具栏。通过该工具栏，用户可以进入写信和管理员界面，同时工具栏中还提供了进入用户各个邮箱帐户的入口，以及进入草稿箱、已删除邮件等信箱的入口。

图 3 发邮件界面

在发邮件界面中（如图 3 所示），用户可以直接将写好的邮件发送到收件人信箱或者保存为草稿。界面的右侧显示了用户添加的联系人列表，用户可以通过直接单击联系人姓名来将其快速加入到收件人当中，而不需要手动输入对方的邮箱地址，这样不仅减少了用户输入出错的几率且为用户提供了方便。

在邮件列表界面中（如图 4 所示），系统用列表的形式展

示了用户接收到的邮件。其中第一列显示邮件发件人的名称或者是邮箱地址中的用户名，第二列显示邮件的标题以及摘要，最后一列显示邮件发送的日期。每一页显示的邮件数目可以通过系统的配置文件进行设定。

图 4　邮件列表界面

在此界面中，用户可以对多封邮件进行删除或者彻底删除操作。被删除的邮件将被移入已删除信箱，用户进入已删除信箱可对该邮件进行恢复操作；彻底删除的邮件将直接从数据库中删除且不可恢复。同时，用户还可以将多封新邮件标记为已读状态。

6　结语

经过前期的调研，中期的开发和后期的测试，系统实现了用户基本信息管理，用户邮箱帐户管理，邮件的接收、发送，草稿邮件管理和联系人管理等功能，并且通过单元测试保证了系统各核心部件功能的正确运行，确保本系统是可用、完整、稳定且安全的。

参考文献

[1] Mike Cantelon / TJ Holowaychuk. Node.js in Action [M] . Manning Press. 2011.

[2] Kristina Chodorow / Michael Dirolf. MongoDB: The Definitive Guide [M] . O'Reilly Media. 2011.

[3] Nicholas C.Zakas. 丁琛，赵泽欣，译. 高性能 JavaScript [M] . 北京: 电子工业出版社，2011.

[4] 郭欣. 构建高性能 Web 站点 [M] . 北京：电子工业出版社，2009.

[5] Nicholas C.Zakas. 李松峰，曹力，译. JavaScript 高级程序设计 [M] . 北京: 人民邮电出版社，2010.

[6] World Wide Web Consortium. RFC822: Standard for ARPA Internet Text Messages [EB/OL] . http://www.w3.org/Protocols/rfc822/.

（收稿日期：2012-10-07）

（上接第 52 页）

T = 恢复单位普通空间的平均时间

p = 快速恢复进行特殊处理所花费的时间

S = 只读表分区的表空间的占用空间

s = 剩下的需要恢复空间

a = 用普通的方法完全恢复数据库的时间

r = 用我们的方法恢复到可用状态的时间

那么

$a = (s+S)\ t + sT = s\ (t+T) + St$

$r = st + sT + p = s\ (t+T) + p$

从上式可以看出当 $p<St$ 时，采用我们的方法是较节省时间的，事实上，一般情况下，经过多年运行的数据库系统，由于历史数据的原因，St 远远大于 p。

8　结语

实践证明，加快 AFC 数据库的备份恢复性能，对轨道交通运营企业来说既是缩短 AFC 突发数据库故障引起运营系统相关 AFC 业务停止服务时间的有效方法，也是提高系统日常响应速度的有力措施。未雨绸缪，针对这些具体的影响业务运营的异常场景，提出相应的对策，必将为企业数据库的平稳运行打下坚实的基础。由于基本没有利用 AFC 的特殊性，故除了适用于 AFC 线路中心数据库外，对于其他行业的数据库的维护管理也具有一定的借鉴意义。

参考文献

[1] 杨玉娟. 地铁 AFC 系统数据库设计、维护和优化 [J] . 铁路计算机应用，2011，(03) .

[2] 张云帆. Oracle 数据库备份与恢复策略 [J] . 计算机工程，2009，(08) .

[3] 王健. Oracle 数据库的备份与恢复策略研究 [J] . 现代情报, 2007，(04) .

[4] 刘恒学. 让 Oracle 的范围分区表按自然方式自动维护分区 [J] .

[5] 王茂林，蒲全武，郭伟. 轨道交通 AFC 系统数据库容灾系统方案设计 [J] . 铁路计算机应用，2010，(02).

[6] 包光磊. 临危不惧：Oracle 11g 数据库恢复技术 [M] .北京：电子工业出版社，2012.

[7] 哈特，弗里曼，江玲玲. Oracle Database 10g RMAN 备份与恢复 [M] . 北京：清华大学出版社，2008.

（收稿日期：2013-03-04）

基于 FTP 与 HTTP 模式手机控制电脑程序的编写

倪慧刚

摘　要：用 Basic 和 Pascal 语言设计一款安卓手机远程指令系统，并以小米手机测试通过，运行良好。

关键词：FTP 协议；HTTP 协议；安卓手机；远程指令

1　引言

随着智能手机的不断普及，手机端与 PC 之间的通信软件越来越多，百度下 “手机控制电脑” 大名鼎鼎的有 Mobile CC Teamview 等，笔者试用过这些软件，它们都能在手机屏幕上显示远程的电脑屏幕，用手指来代替鼠标操作，完成在手机屏幕上操作远程电脑，显得非常方便，其优点是直观显示桌面，实时操作电脑，但由于手机屏幕较小，观看与操作上不是特别顺手，因为是实时传播电脑画面图片，在 3G 收费用户看来，会产生较大的流量，如果屏幕比例改为 1:1，操作过程中的桌面来回拖动定位也不太方便。

接下来和大家动手打造一款自己的手机控制电脑的小程序，本程序是基于 FTP 与 HTTP 模式的，撇开了电脑桌面、鼠标等图像的传输，在流量上更加节约，功能上虽然不及上面提到的专业类软件，但也还算可以，当然最关键的体会自己动手的乐趣。

图 1 是手机端程序的界面。

图 1　手机端程序界面

2　开发工具

手机端使用 Basic4android 可以到官方网站下载，如图 2 所示。

图 2　Basic4android 界面

只要会 Basic 语言的朋友，马上可以上手，操作十分方便。PC 端程序使用 Delphi7，非常经典的老牌开发工具，如图 3 所示。

图 3　Delphi7 界面

Basic4android 的开发环境配置可以根据官方网站的帮助提示，Delphi7 与组件的安装比较方便，不再赘述。

3　程序原理

本系统是基于 FTP 与 HTTP 模式的，手机端程序利于 Basic4andriod 提供的 ABFTP 组件库中的 FTP 上传功能，将指令以 “年月日-时分-命令 00.txt ” 的格式作为文件名上传到指定 FTP 文件夹中，PC 端程序则是利用 webbrowser 组件定时访问一个能得到当前 FTP 文件夹中文件名的 ASP 程序，然后，取到网页内容，经处理过滤，对上传的指令文件名进行分离，

得到年月日时分与命令两部分，最后与当前系统当前时间时行比对，当上传的年月日时分与系统时间一致时，便执行相应的指令，而所谓的指令就是 PC 端指定文件夹的是一些批处理或第三方程序。所以，本系统的一个特点是第三方程序与批处理可以自行修改或增加，使得软件的扩展功能更加大。

4 代码解析

手机端的界面在设计时如图 4 所示，基本和 VB6 等 IDE 相同。

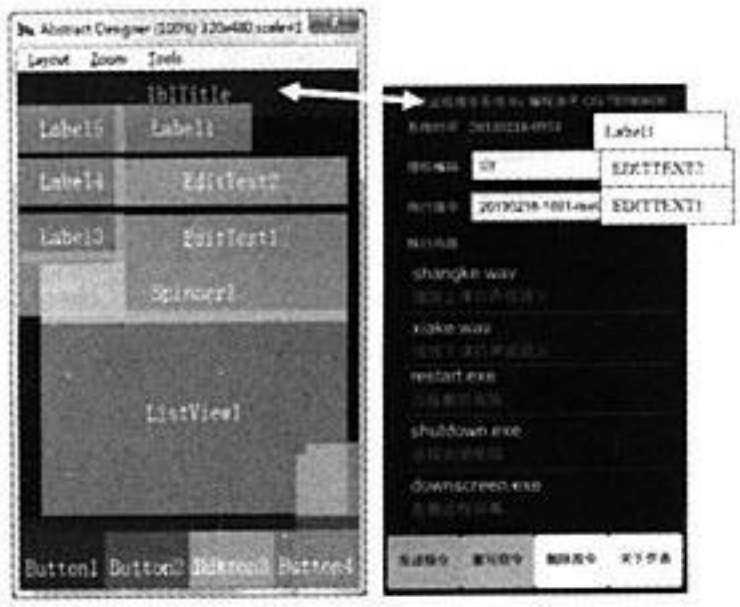


图 4 手机端界面设计

不难发现，界面和编译好直接运行时基本一样，组件的显示效果大家可以对照参看。

Label1 是用于显示当前时间的，精确到分，在 Basic4android 软件中的源码，如图 5 所示。

图 5 Basic4android 中输入代码

首先定义一个 Timer1_Tick，相当于在 Delphi 中放一个 timer 组件，其中调用的过程中的代码就是让 label1 显示时间。同时，必须设置好 timer 组件的一些属性和 listview1 的内容。在 Activity_Create 过程中进行设置如下：

```
Sub Activity_Create(FirstTime As Boolean) ' 此过程表示程序
' 初始化就执行代码，
    Activity.LoadLayout("MainLayout")  ' 指定程序层，
     ' Timer1.Initialize("Timer1", 1000) ' 指定 timer1 的时钟频
' 率，即一秒一次
    Timer1.Enabled = True  ' 使 timer 组件生效，
              ' 下面是定义内容，可按格式根据自己需要来设定
' listview1 的下拉内容
     ListView1.AddTwoLines ("shangke.wav","播放上课铃声
或音乐")
ListView1.AddTwoLines ("xiake.wav","播放下课铃声或音乐")
      ListView1.AddTwoLines ("restart.exe", "远程重启电脑")
ListView1.AddTwoLines ("shutdown.exe","远程关闭电脑")
      ListView1.AddTwoLines ("downscreen.exe","查看远程屏
幕....")
End Sub
```

为了方便发送指令，只需点击 listveiw1 相应的内容，就能自动将指令加入到 EDITTEXT1 上，代码如下：

```
Sub ListView1_ItemClick (Position As Int, Value As Object)
  Timer1.Enabled = False    ' 先让时钟组件暂停下来，
     Edittext1.Text = label1.Text&"-"&Value    'Value 的值就
' 是点击 listview1 的内容
End Sub
```

接下去就是最重要的功能了，就是 Button1 发送指令按键的代码如下：

```
Sub Button1_Click     '' 发送指令按钮
  Dim sdRoot ,FTPRoot, myfile() As String
  sdRoot = File.DirRootExternal&"/"  ' 定义 SD 卡根目录
   FTPRoot = "/web/51dkxt/"&edittext2.text&""     ' 定义FTP
' 文件夹位置
  If File.Exists(sdRoot,"nihg.txt") Then   'nihg.txt 为授权文件，
' 放在 SD 卡根目录下
     edittext2.text=File.ReadString (sdRoot,"nihg.txt")  ' 显示
' 授权编码内容
       myFTP.ABFTPConnect("180.178.0.171",21,"111","222")
                'FTP 连接，格式为服务器，端口，用户名，密码
     If myFTP.ABFTPConnected = True Then  '("成功连接到
服务器", "")
myFTP.ABFTPUpload(sdRoot,edittext1.Text&"00.txt",FTPRoot,
edittext1.Text&"00.txt")  ' 上传到 FTP 上，格式为 本地文件夹，
' 文件名，FTP 文件夹，文件名
      If myFTP.ABFTPLastError <> "" Then
         Msgbox(myFTP.ABFTPLastError,"") ' 如果上传错，
' 显示出错代码
File.Delete (sdRoot,edittext1.Text&"00.txt")   ' 删除本地文件
    Else
 Msgbox("恭喜！ 远程指令已成功下达...","") ' 上传成功
          File.Delete (sdRoot,edittext1.Text&"00.txt")
                  Select Msgbox2 (" 是否查看远程指令列
表?", "", "现在查看", "不查看了","", Null)
                  ' 提示是否查看已经下达的指令
    Case -1  ' 选择 现在查看 button3.visible=True ' 删除
' 指令 按钮可见
                  ListView2.Clear
                  spinner1.Clear
       ListView2.Visible =True  ' 显示远程指令列表
       FTPRoot = "/web/51dkxt/"&edittext2.text&""
       myfile=myFTP.ABFTPListFiles (FTPRoot)   ' 得到
' FTP 文件列表 spinner1.Add ("点击下拉框，选择要删除的指
' 令")   ' 列表加到 spinner1 组件上
```

```
For a= 0 To myfile.Length -2  -2 是为了不显示多余文件
ListView2.AddSingleLine (myfile(a))
                spinner1.Add (myfile(a))
                Next
```

当循环结束后，取到所有 FTP 文件名后，spinner1 的效果以下点击后下拉内容如图 6 所示。

图 6　spinner1 的效果

```
Case 0    ′选择不查看,就不做什么
                End Select
                        End If
                    Else
                      Msgbox("连接服务器失败", "")
                      End If
                            Else
          Msgbox("请联系作者获取授权编码", "")
        ′如果不存丰 nihg.txt 文件,就提示没有授权
                            End If
                        End Sub
```

删除指令的按钮事件如下：

```
Sub Button3_Click   ″删除服务器上的指令
 spinner1.Visible =True  ′出现菜单,只是让 spinner1 显示
 End Sub
```

至此，完成了指令发送的任务，手机端程序的主要功能基本完成了，下面再来看一下删除指令的代码，如图 6 中的右图下拉框，spinner1 的相应代码如下：

```
Sub Spinner1_ItemClick (Position As Int, Value As Object)
Dim ftproot As String
ftproot = "/web/51dkxt/"&edittext2.text&""
Select  Msgbox2 (spinner1.GetItem (spinner1.SelectedIndex ),
"", "删 除", "不删除","", Null)      ′当选择其中一行时,出现提示
                            Case -1  ′进入删除指定指令
 myFTP.ABFTPDeleteFile (ftproot,spinner1.GetItem (spinner1.
SelectedIndex ))
 ′删除 FTP 上的文件
If myFTP.ABFTPLastError <> "" Then
    Msgbox("远程指令删除失败...","")

     Else
     Msgbox("远程指令已删除...","")
                                End If
Case 0        ′不删除指定指令
                                End Select
        spinner1.Visible =False   ′删除后,把 spinner 隐藏
     button3.Visible =False    ′删除后,把删除按钮也隐藏
End Sub
```

至此完成了本系统的手机端功能，下面来看看 PC 端的主界面，如图 7 所示。

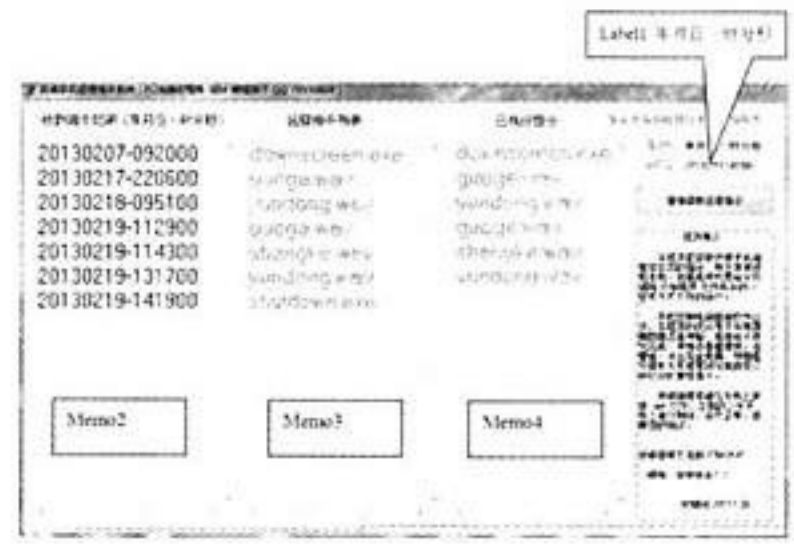


图 7　PC 端的主界面

PC 端程序用：

```
label1.Caption  := FormatDateTime ('yyyymmdd-hhmmss', now);
′来得到当前系统时间。
```

先构造一个 http://www.shengzuan100.com/51dkxt/123/9999999999999.asp 文件，其内容为：

```
<%
′创建一个 FileSystemObject 对象的事例
        Set  MyFileObject =Server.CreateObject ("Scripting.
FileSystemObject")
path=server.mappath("./") ′当前目录下 folder 文件夹目录
′创建一个 Folder 对象
Set MyFolder=MyFileObject.GetFolder(path)
For Each thing in MyFolder.SubFolders ′获取子文件夹
 response.write "<p>目录:"&thing
′MyFileObject.DeleteFolder thing 删除文件夹,注意使用
next
′循环显示其中文件
For Each thing in MyFolder.Files
Response.Write("<p>文件:"&thing) ′输出文件路径
Next
%>
```

这个文件的功能是得到 FTP 当前文件夹的文件列表，保存后上传到 Web，访问该文件，将返回网页内容如图 8 所示。

当然，用 webbrowser 组件访问此 ASP 文件时得到文字是在网页上的，得想法子取到这些文本内容：

```
url:=  ′http://www.shengzuan100.com/51dkxt/123/9999999999999.asp′;
webbrowser1.Navigate(url)  ;
          memo1.text: =webbrowser1.OleObject.document.
documentelement.innerText; // 此时,上述内容就在 memo1
//组件中了,这样就能为程序所用了。
```

(下转第 79 页)

颜色相似系数的目标提取与颜色替换

陶 胜

摘 要：探讨利用区域生长进行目标提取，提出一种基于颜色相似系数的区域生长算法，该算法以待判别点与种子点的颜色相似系数大于给定阈值作为生长准则，给出颜色相似系数的定义及计算公式，对提取的目标进行颜色替换，并应用 Matlab 软件编程实现。运行结果表明效果不错。

关键词：颜色相似系数；区域生长；目标提取；颜色替换

1 引言

目标提取是指利用图像的灰度、梯度、纹理、颜色、形状等信息从复杂的背景中提取出感兴趣的目标，使得更高层的图像分析和理解成为可能。在图像处理应用中，已知图像中感兴趣目标的种子点，可以利用区域生长算法将其提取出来。区域生长是一种重要的图像分割方法，它将具有某种相似性质的像素集合起来构成区域。

彩色图像最常用的颜色表示是 RGB 颜色空间，是一种基本的颜色描述方法，但由于 RGB 颜色空间的非均匀性，并不能很好满足颜色视觉特性，而采用其他的近似均匀的颜色空间，存在空间变换复杂、计算繁琐等缺陷，影响了大数据量彩色图像处理的实时性要求。

在 RGB 颜色空间中给出色度饱和度相似系数和亮度相似系数的定义与计算公式，并将它们的加权和作为颜色相似系数。在此基础上，探讨利用区域生长法提取彩色图像中的目标，以待判别点与种子点的颜色相似系数大于给定阈值作为生长准则，对提取的目标进行颜色替换，并应用 Matlab 软件编程实现。

2 颜色相似度

颜色相似度是彩色图像处理的基本运算，它可以用来分析两种颜色之间的差异。一种理想的颜色相似度量方法应该采用基本的 RGB 颜色空间表示颜色，而不必进行复杂的颜色转换过程，同时满足颜色视觉特性。

在 RGB 颜色空间中，颜色向量是用三维矢量来表示的。设有两种颜色用颜色矢量分别表示为 $f=(f_1, f_2, f_3)$，$g=(g_1, g_2, g_3)$，其欧氏距离 $d(f,g)$为：

$$d(f,g)=\sqrt{(f_1-g_1)^2+(f_2-g_2)^2+(f_3-g_3)^2}$$

如果采用两个颜色向量之间的欧氏距离来度量它们之间的差异，并不能很好满足颜色视觉特性。考察 3 个颜色向量 $C1$=(216,216,216)，$C2$= (208,208,208)，$C2$= (224,216,205)，用人眼容易观察出与 $C1$ 比 $C2$ 与 $C3$ 的颜色接近，但 $d(C1,C2)$ = 13.86，$d(C1,C3)$ =13.60，$d(C1,C2)>d(C1,C3)$。

2.1 色度饱和度相似系数

对于两种颜色矢量 $f=(f_1, f_2, f_3)$，$g=(g_1,g_2,g_3)$，这两个颜色矢量之间的夹角余弦可以作为它们之间的相关系数，用 $r(f,g)$ 表示如下：

$$r(f,g)=\frac{f_1g_1+f_2g_2+f_3g_3}{\sqrt{(f_1^2+f_2^2+f_3^2)(g_1^2+g_2^2+g_3^2)}}\quad r(f,g)\in[0,1]。$$

$r(f,g)$ 值越大，相似程度越大，反之亦然。称这个相关系数为色度饱和度相似系数，可直接用于颜色相似性的分析计算中。

2.2 亮度相似系数

由两种颜色的色度饱和度相似系数计算公式可以看出，两种颜色的色度饱和度相似系数为 1 时，表示两种颜色各个分量对应成比例，因此仅色度饱和度相似系数还无法完全表示出两种颜色的相似性程度，需要进一步比较两种颜色的亮度。可定义一个比较两种颜色 f 和 g 之间的亮度相似系数 $k(f,g)$：

$$k(f,g)=1-\frac{|f_1+f_2+f_3-g_1-g_2-g_3|}{C}$$

式中，C=3×255=765，$k(f,g)\in[0,1]$ 当两种颜色亮度相同时，系数 $k(f,g)$=1，当两种颜色亮度差增大时，$k(f,g)$值减小。

2.3 颜色相似系数

从人眼颜色视觉特性来分析，色度和饱和度的区别是主要的，决定了颜色的主要特性。所以在上述的两个系数中，色度饱和度相似系数代表了颜色相似性的主要方面，而亮度相似系数代表颜色相似性的次要方面。综合考虑这两个系数，可以分别对两个系数加权，并设定权值来调整两个系数在颜色相似性度量中的重要程度，这样得到了一个综合的颜色相似性度量系数 $s(f,g)$：

$$s(f,g)=a_1r(f,g)+a_2k(f,g)$$

式中，$a_2+a_2=1$，$a_1>0$，$a_2>0$，$a_1>a_2$。

例如在RGB空间中，设有两种颜色分别为*Color*1=(216,8,24)，*Color*2 (240,16,16)，则它们的色度饱和相似度为0.999，而亮度相似系数为0.969，所以两者的颜色相似系数为$0.999a_1+0.969a_2$。实际应用可取$a_1=0.85$，$a_2=0.15$，则s (*Color*1, *Color*2) =0.994。判断这两种颜色是否相似，可根据s (f,g)的数值来判定：选取阈值$\tau\in[0,1]$，如果$s(f,g)>\tau$，则认为两种颜色相似，否则判定不相似。实际应用中，τ可取为0.985，所以可以认为这两种颜色相似。

3 目标提取

要将彩色图像中的目标提取出来，其实质就是将待提取的目标和其他部分分割开来，可以应用区域生长法来实现。区域生长的基本思想是将具有相似性质的像素集合起来构成区域，首先确定种子像素作为生长的起点，然后根据某种事先确定的生长或相似准则，在种子像素周围领域中寻找与种子像素有相同或相似性质的像素，并将这些像素合并到种子像素所在的区域中。将这些新像素作为新的种子像素继续进行上述过程，直到没有满足条件的像素可被包括进来，这样一个区域就长成了。

由此可见，区域生长算法主要有3点：生长种子点的确定、区域生长的条件和区域生长停止的条件。种子点的个数根据具体的问题可以选择一个或者多个，并且根据问题的不同可以采用完全自动确定或者人机交互确定。而区域生长的条件是基于上述颜色相似性判定规则来确定，即：种子像素与它周围的未被标记的4邻域像素进行比较，任何满足颜色相似性判定条件的邻域像素被指定到生长区域中，并给予标记。对于每个分配到生长区域中的像素作为新的种子点，不断地重复与邻域像素颜色相似性判定的步骤，直到没有未被标记的种子点为止，这就是区域生长停止的条件。

4 颜色替换

假设从彩色图像中提取出来的目标为*Object*，P为*Object*的任意一点，P点的颜色表示为(r,g,b)，将其替换成颜色(r', g',b')。

4.1 简单替换

将P点颜色的3个分量中的两个分量互换，另一个保持不变，实现简单替换。例如：

r'=g,g'=r,$b'=b$

也可以是3个分量之间轮换，例如：

r'=g,g'=b,$b'=r$

4.2 按指定颜色替换

假设指定颜色为*Color*= (R,G,B)，将P点的颜色(r,g,b)替换成(r',g',b')，指定颜色*Color*的亮度为$\frac{R+G+B}{3}$，而P点的颜色亮度为$\frac{r+g+b}{3}$，要保持颜色替换前后亮度一致，可以按如下公式进行：

$$r'=\frac{(r+g+b)*r}{R+G+B}$$

$$g'=\frac{(r+g+b)*g}{R+G+B}$$

$$b'=\frac{(r+g+b)*b}{R+G+B}$$

注：如果计算结果r'大于255，则将r'取为255。对于g'和b'，也类似处理。

5 实验结果

采用Matlab7作为开发工具，在Windows XP平台下实现了提取彩色图像中的目标和颜色替换，效果如图1-图4所示。

图1 彩色原图

图2 提取彩色图像中的衣服对象

图3 衣服颜色替换成绿色

图4 衣服颜色替换成紫色

6 结语

给出了颜色相似系数的定义及计算公式，提出了一种基于颜色相似系数的区域生长算法，应用该区域生长法进行目标提取，并实现了对提取的目标进行颜色替换。通过仿真实验，得出如下结论：

(1) 颜色相似性判定规则根据计算颜色相似系数来确定，比根据计算颜色之间的欧氏距离来确定，提取目标的效

(下转第90页)

基于遗传算法的地图四色问题研究

刘 烽

摘 要：尝试利用遗传算法求解地图四色问题，合理设计了算法的编码、适应度函数和交叉变异算子，通过仿真实验，验证了遗传算法对于求解地图四色问题具有较好的效果。

关键词：遗传算法；地图四色；种子填充

1 问题描述

地图四色问题又称四色猜想、四色定理，是世界近代三大数学难题之一。四色问题的内容是："任何一张地图只用四种颜色就能能使具有共同边界的国家着上不同的颜色。"四色问题虽然解决了一张地图的着色仅需四种颜色，但并没有给出具体的着色算法。目前，求解地图四色着色的方法主要有：一般回溯法、启发式搜索算法。鉴于遗传算法较好的寻优能力，算法实现简单，下文以某地图为例，利用该算法解出各区域的填充颜色。

如图 1 所示，给定红色、绿色、蓝色、粉红色 4 种颜色，要求用这 4 种颜色对图中区域进行着色，使得任何相邻区域的颜色各不相同。

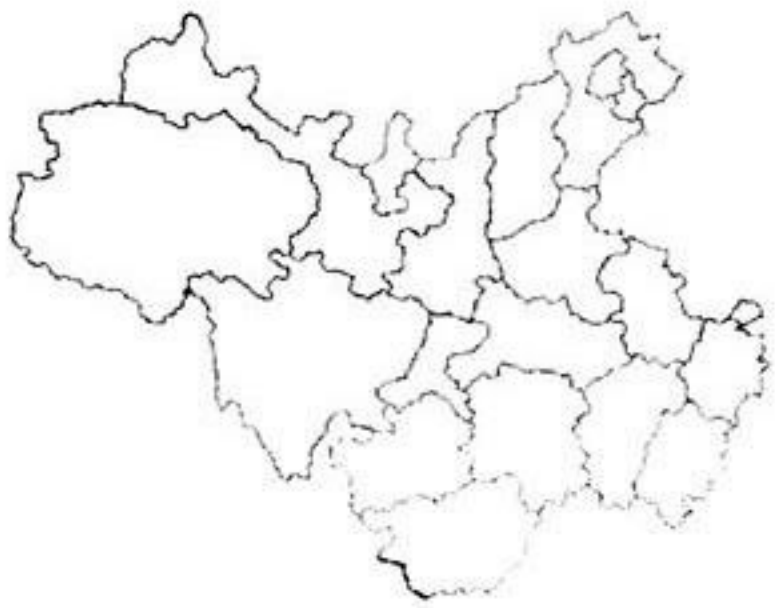


图 1 示例地图

2 算法设计思路

2.1 染色体编码

如果用 1、2、3、4 分别代表红色、绿色、蓝色、粉红色 4 种颜色，图 1 各区域的填充方法则是由 1、2、3、4 组成的长度为 20 位（图中共有 20 个区域）的一串编码，这一串编码即可作为染色体，其中的基因代表每个区域的填充颜色，基因的序号代表每个区域，各区域的序号按事先定义好的序号设定。

2.2 选择、交叉、变异操作

2.2.1 选择

遗传算法的选择操作通常有：轮盘赌选择法、锦标赛选择和精英选择法等，为使算法易于实现，这里使用轮盘赌选择法。

2.2.2 交叉

交叉操作通常有单点交叉、多点交叉等，本文采用单点交叉，交叉点的位置由程序随机生成。

2.2.3 变异

在遗传算法中，变异的作用主要是增加种群的多样性，使算法能跳出局部最优。文中的变异操作按如下方法进行：即随机生成一个染色体的变异位置，从（1，2，3，4）中随机生成一位取代原有基因。

2.3 适应度函数

适应度用于评价染色体的优劣程序，针对图 1 中四色填充的特点，将相邻区域着相同颜色的冲突次数的倒数作为染色体的评价函数，需要说明一点的是，为了避免最优个体出现 1 除以 0 的情形，冲突次数的记数初值定为 1，因此最优个体对应的适应度为 1。

3 程序实现

算法实现采取 VC6.0 环境下 MFC 的编程，程序在 WinXP 系统中调试通过。本程序为基于对话框类型程序，示例地图采用一幅像素为 624*512 大小的 bmp 图像。程序界面设计如图 2 所示。

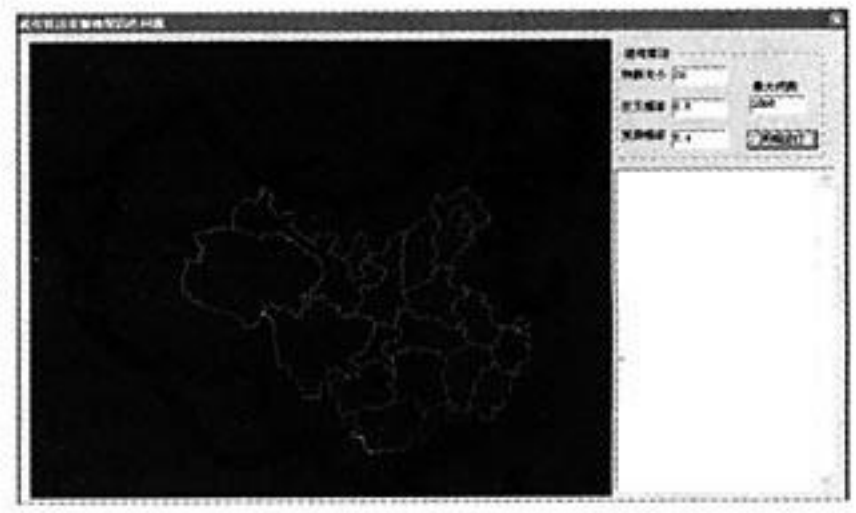


图 2 程序界面设计

程序的核心模块主要有初始化模块、遗传算法模块、地图背景处理及填充模块：

```
/
*********************************************************/
/*                  对话框初始化模块                     */
/
*********************************************************/
BOOL CMyDlg::OnInitDialog()
    {
        CDialog::OnInitDialog();
            // IDM_ABOUTBOX must be in the system
// command range.
        ASSERT ((IDM_ABOUTBOX & 0xFFF0) ==
IDM_ABOUTBOX);
        ASSERT(IDM_ABOUTBOX < 0xF000);

        CMenu* pSysMenu = GetSystemMenu(FALSE);
        if (pSysMenu ! = NULL)
        {
            CString strAboutMenu;
            strAboutMenu.LoadString(IDS_ABOUTBOX);
            if (! strAboutMenu.IsEmpty())
            {
        pSysMenu->AppendMenu(MF_SEPARATOR);
        pSysMenu ->AppendMenu (MF_STRING,
IDM_ABOUTBOX, strAboutMenu);
            }
        }
            // Set the icon for this dialog.  The framework
// does this automatically
            //  when the application's main window is not a
// dialog
      SetIcon(m_hIcon, TRUE);            // Set big icon
        SetIcon(m_hIcon, FALSE);        // Set small icon
      // TODO: Add extra initialization here
        //填满邻居矩阵的另一半
        for(int i=0;i<20;i++)
            for(int j=i;j<20;j++)
                if(NeighbourMatrix[i][j]==1)
                    NeighbourMatrix[j][i]=1;
        //种群大小
        m_size=20;
        //交叉概率
        m_CrossProbability=0.8;
        //变异概率
        m_MutateProbability=0.4;
        //算法代数
        m_GENERATION=2000;
        UpdateData(false);
          return TRUE;  // return TRUE  unless
// you set the focus to a control
    }
/*********************************************************/
/*                  遗传算法求解模块                     */
/*********************************************************/
void CMyDlg::GA()
{
    int SIZE;
double CrossProbability,MutateProbability;//交叉和变异概率
    long GENERATION;        //进化代数
    UpdateData(true);
    SIZE=m_size;
    CrossProbability=m_CrossProbability;
    MutateProbability=m_MutateProbability;
    GENERATION=m_GENERATION;
    int lable=0;     //临时存放的数组标识
    int i,j,k,l;
    double sum=0,sum1=0,sum2=0;
    double temp=0;
    int temp2=0;
    int num1,num2;
    int **allmatrix;
    allmatrix=new int *[SIZE];
    for(i=0;i<SIZE;i++)
    {
        allmatrix[i]=new int[20];
    }
    int **allmatrix_temp;
    allmatrix_temp=new int *[SIZE];
    for(i=0;i<SIZE;i++)
    {
        allmatrix_temp[i]=new int[20];
    }
    double *fitness;                    //录入 B 数组
    fitness=new double[SIZE];
    double best_fitness;
    int best_number;
    CString tempstring;
    srand((unsigned)time(NULL));
    //种群初始化
    for(i=0;i<SIZE;i++)
    {
        for(j=0;j<20;j++)
        {
            allmatrix [i] [j] =int (4*double (rand ())/
(RAND_MAX+1));
        }
    }
    for(l=0;l<=GENERATION;l++)
    {
        sum=0;
        sum1=0;
        sum2=0;
```

```
        //计算个体适应度
        for(i=0,fitness[i]=1;i<SIZE;i++,fitness[i]=1)

        {
            for(j=0;j<20;j++)
                for(k=j+1;k<20;k++)
                    if (NeighbourMatrix [j] [k] ==
1&&allmatrix[i][j]==allmatrix[i][k])
                        fitness[i]++;
        }
        for(i=0;i<SIZE;i++)
            fitness[i]=1/fitness[i];
        best_fitness=fitness[0];
        best_number=0;
        for(i=1;i<SIZE;i++)
        {
            if(fitness[i]>best_fitness)
            {
                best_fitness=fitness[i];
                best_number=i;
            }
        }
        if(I%20==0)
        {
            tempstring.Format("%d",I);
            info+="第";
            info+=tempstring;
            info+="代中,最佳适应值为:";
            tempstring.Format("%f",best_fitness);
            info+=tempstring;
            info+="\r\n";
            ShowWindow();
        }
        if(best_fitness==1)
        {
            for(i=0;i<20;i++)
            {
                color[i]=allmatrix[best_number][i]+1;
            }
            tempstring.Format("%d",I);
            info+="第";
            info+=tempstring;
            info+="代中,最佳适应值为:";
            tempstring.Format("%f",best_fitness);
            info+=tempstring;
            info+="\r\n";
            ShowWindow();
            info.Empty();
            MultipleRegionGrow(*bmpTwo);
            I=GENERATION+2;          //跳出循环
            break;
        }
        for(i=0;i<SIZE;i++)
            sum+=fitness[i];
        for(lable=0;lable<SIZE;lable=lable+2)       //一次完
//整的选择杂交过程
        {
            temp=double(rand())/RAND_MAX;
            i=0;
            sum1=fitness[0];
            while(double(sum1/sum)<temp)//轮盘赌法选
//择交叉的母代 num1
            {
                i++;
                sum1+=fitness[i];
            }
            num1=i;

            temp=double(rand())/RAND_MAX;
            i=0;
            sum2=fitness[0];
            while(double(sum2/sum)<temp)//轮转法选择
//交叉的母代 num2
            {
                i++;
                sum2+=fitness[i];
            }
            num2=i;
            temp2 =int (20*double (rand ())/(RAND_MAX +
1));  //随机生成交叉点位置
                if (double (rand ())/RAND_MAX <
CrossProbability)      //是否交叉
            {
                for(i=0;i<temp2;i++)
                {
            allmatrix_temp[lable][i]=allmatrix[num1][i];
            allmatrix_temp[lable+1][i]=allmatrix[num2][i];
                }
                for(i=temp2;i<20;i++)
                {
            allmatrix_temp[lable][i]=allmatrix[num2][i];
            allmatrix_temp[lable+1][i]=allmatrix[num1][i];
                }
            }
            else
            {
                for(i=0;i<20;i++)
                {
        allmatrix_temp[lable][i]=allmatrix[num1][i];
        allmatrix_temp[lable+1][i]=allmatrix[num2][i];
                }
            }
        }
```

```
        for(i=0;i<SIZE;i++)
        {
                if (double (rand ())/RAND_MAX <
MutateProbability)
                {
                        temp2 =int (20*double (rand ())/
(RAND_MAX+1));    //随机生成变异点位置
                allmatrix_temp [i] [temp2] =int (4*double
(rand())/(RAND_MAX+1));
                }
        }

        for(i=0;i<SIZE;i++)
        {
                for(j=0;j<20;j++)
                {
                        allmatrix[i][j]=allmatrix_temp[i][j];
                }
        }
}
if(l==GENERATION+1)
{
        info+="无解！ \r\n";
        ShowWindow();
        info.Empty();
}
    /*********************************************/
    /*                地图背景处理模块                */
    /*    读取地图信息，并将真彩色地图进行二值化处理
*/
   /*********************************************/
    void CMyDlg::Open()
    {
        BYTE * pBuffer;

        CFile bmp("地图.bmp",CFile::modeRead);
        bmp.Seek(54 ,CFile::begin);
        pBuffer=new BYTE[624*512*3];
        bmp.Read(pBuffer,624*512*3);
        bmp.Close();

        CClientDC dc(this);
        int r,g,b;
        int i,j;
        for(i=0;i<512;i++)
        {
            for(j=0;j<624;j++)
            {
                b=*pBuffer;
                g=*(pBuffer+1);
                r=*(pBuffer+2);
                pBuffer=pBuffer+3;

                if((r+g+b)/3>50)
                    bmpTwo[i][j]=255;
                else
                    bmpTwo[i][j]=0;

                dc.SetPixel   (j  +11,10  +512  -i,RGB
(bmpTwo[i][j],bmpTwo[i][j],bmpTwo[i][j]));
                    bmpTwo[i][j]=255-bmpTwo[i][j];
                }
            }
        }
        /
*********************************************/
        /*                地图背景处理模块                */
        /*    便于后续填充算法，将地图轮廓线进行细化处理
*/
                                                        /
*********************************************/
        void CMyDlg::Thining()
        {
            int i=0,j;
            int flag = 1;
            BYTE **bmpThin;
            bmpThin = new BYTE *[512];
            for(i=0;i<512;i++)
                bmpThin[i] = new BYTE[624];

            for(i=0;i<512;i++)
                for(j=0;j<624;j++)
                    //初始化
                    bmpThin[i][j]= 0;
                //flag=0 时迭代结束
                while(flag==1)
                {
                    flag=0;
                    for( i=2;i<512-2;i++)
                    {
                        for(j=2;j<624-2;j++)
                        {
                if(bmpTwo[i-1][j-1]==0xff//模板 a
            && bmpTwo[i-1][j+1]==0
                && bmpTwo[i][j-1]==0xff
                        && bmpTwo[i][j-0]==0
                        && bmpTwo[i][j+1]==0
                        && bmpTwo[i][j+2]==0
                && bmpTwo[i+1][j-1]==0xff
            && bmpTwo[i+1][j+1]==0)
                        {
                            bmpThin[i][j]=0xff;
                            flag=1;
```

```
                    continue;
                }

                if(bmpTwo[i-1][j-0]==0
//模板 b
        && bmpTwo[i-1][j+1]==0
            && bmpTwo[i][j-1]==0xff
                && bmpTwo[i][j-0]==0
                && bmpTwo[i][j+1]==0
                && bmpTwo[i][j+2]==0
&& bmpTwo[i+1][j-1]==0xff
            && bmpTwo[i+1][j-0]==0xff)
            {
                bmpThin[i][j]=0xff;
                flag=1;
                continue;
            }

if(bmpTwo[i-1][j-1]==0xff//模板 c
&& bmpTwo[i-1][j-0]==0xff
    && bmpTwo[i][j-1]==0xff
                && bmpTwo[i][j-0]==0
                && bmpTwo[i][j+1]==0
                && bmpTwo[i][j+2]==0
&& bmpTwo[i+1][j-0]==0
    && bmpTwo[i+1][j+1]==0)
            {
                bmpThin[i][j]=0xff;
                flag=1;
                continue;
            }

if(bmpTwo[i-1][j-1]==0xff//模板 d
&& bmpTwo[i-1][j-0]==0xff
&& bmpTwo[i-1][j+1]==0xff
&& bmpTwo[i][j-0]==0
&& bmpTwo[i+1][j-1]==0
&& bmpTwo[i+1][j-0]==0
&& bmpTwo[i+1][j+1]==0)
            {
                bmpThin[i][j]=0xff;
                flag=1;
                continue;
            }

                if(bmpTwo[i-1][j-1]==0
//模板 e
        && bmpTwo[i-1][j+1]==0xff
                && bmpTwo[i][j-1]==0
                && bmpTwo[i][j-0]==0
&& bmpTwo[i][j+1]==0xff
    && bmpTwo[i+1][j-1]==0
        && bmpTwo[i+1][j-0]==0xff
        && bmpTwo[i+1][j+1]==0xff)
            {
                bmpThin[i][j]=0xff;
                flag=1;
                continue;
            }

                if(bmpTwo[i-1][j-1]==0
//模板 f
        && bmpTwo[i-1][j+1]==0xff
                && bmpTwo[i][j-1]==0
                && bmpTwo[i][j-0]==0
            && bmpTwo[i][j+1]==0xff
            && bmpTwo[i+1][j-0]==0xff
            && bmpTwo[i+1][j+1]==0xff)
            {
                bmpThin[i][j]=0xff;
                flag=1;
                continue;
            }
if(bmpTwo[i-1][j-0]==0xff//模板 g
    && bmpTwo[i-1][j+1]==0xff
        && bmpTwo[i][j-1]==0
    && bmpTwo[i][j-0]==0
    && bmpTwo[i][j+1]==0xff
    && bmpTwo[i+1][j-0]==0)
            {
                bmpThin[i][j]=0xff;
                flag=1;
                continue;
            }
if(bmpTwo[i-2][j-0]==0          //模板 h
    && bmpTwo[i-1][j-1]==0
        && bmpTwo[i-1][j-0]==0
    && bmpTwo[i-1][j+1]==0
    && bmpTwo[i][j-0]==0
&& bmpTwo[i+1][j-1]==0xff
&& bmpTwo[i+1][j-0]==0xff
    && bmpTwo[i+1][j+1]==0xff)
            {
                bmpThin[i][j]=0xff;
                flag=1;
                continue;
            }
        }
    }
for(i=0;i<512;i++)
    {
        for(j=0;j<624;j++)
      if(bmpThin[i][j]==0xff)
                    bmpTwo[i][j]=0xff;
 }
}
```

```
for( i=2;i<512-2;i++)          //第二次串行细化
{
   for(j=2;j<624-2;j++)
   {
      //缩小后的模板 a
      if(bmpTwo[i-1][j-1]==0xff
                  && bmpTwo[i-1][j+1]==0
                  && bmpTwo[i][j-1]==0xff
                  && bmpTwo[i][j-0]==0
                  && bmpTwo[i][j+1]==0
                  && bmpTwo[i+1][j-1]==0xff
                  && bmpTwo[i+1][j+1]==0)
{
                  bmpTwo[i][j]=0xff;
                  flag=1;
                  continue;
}
//缩小后的模板 h
if(bmpTwo[i-1][j-1]==0
                  && bmpTwo[i-1][j-0]==0
                  && bmpTwo[i-1][j+1]==0
                  && bmpTwo[i][j-0]==0
                  && bmpTwo[i+1][j-1]==0xff
                  && bmpTwo[i+1][j-0]==0xff
                  && bmpTwo[i+1][j+1]==0xff)
      {
                  bmpTwo[i][+j]=0xff;
                  flag=1;
                  continue;
      }
        }
}
   delete []bmpThin;
   CClientDC dc(this);

   //     for(i=0;i<512;i++)
   for(i=511;i>=0;i--)
   {
         for(j=0;j<624;j++)
         {
               if(bmpTwo[i][j]==0xff)
            dc.SetPixel(j+11,512+10-i,RGB(255,255,255));
               else
                     dc.SetPixel(j+11,512+10-i,RGB(0,0,0));
            }
      }
}
/*************************************************/
/*                颜色填充模块                   */
/*    并行区域增长算法，用于对各区域进行种子填充
*/
/*************************************************/
```

```
void CMyDlg::MultipleRegionGrow(BYTE *bwImage)
{
      static int nDx[]={-1,0,1,0};
      static int nDy[]={ 0,1,0,-1};
      // 定义堆栈的起点和终点
      // 当 nStart=nEnd, 表示堆栈中只有一个点
      long int nStart=0 ;
      long int nEnd=0   ;
      //     p_bw=new BYTE(512*624);
      int *GrowRegionx = new int [512*624];
      int *GrowRegiony = new int [512*624];
      // 当前正在处理的像素
      int nCurrx ;
      int nCurry ;
      // 循环控制变量
      int k ;
      // 图像的横纵坐标,用来对当前像素的 4 邻域进行遍历
      int xx;
      int yy;
      CClientDC dc(this);
      memset(p_bw,0,sizeof(BYTE)*319488);
      nStart=0;
      nEnd=0;
      // 把种子点的坐标压入栈
      for(int i=0;i<20;i++)
      {
            //千位数表示颜色
            GrowRegionx[nEnd] = center_y[i]+color[i]*1000;
GrowRegiony[nEnd] = center_x[i];
            nEnd++;
      }
      nEnd--;
      while (nStart<=nEnd)
      {
            // 当前种子点的坐标
            nCurrx = GrowRegionx[nStart];
            nCurry = GrowRegiony[nStart];
            // 对当前点的 4 邻域进行遍历
            for (k=0; k<4; k++)
            {
                  // 4 邻域像素的坐标
                  xx = nCurrx+nDx[k];
                  yy = nCurry+nDy[k];

                  // 判断像素(xx,yy) 是否在图像内部
                  if (   ((xx%1000) < 624) && ((xx%1000)>=0)
&& (yy<512) && (yy>=0)
                        &&p_bw[yy*624+(xx%1000)]==0
                  &&bwImage[yy*624+(xx%1000)]==255)
                  {
                        // 堆栈的尾部指针后移一位
                        nEnd++;
```

```
                // 像素(xx,yy) 压入栈
                GrowRegionx[nEnd] = xx;
                GrowRegiony[nEnd] = yy;
                // 同时也表明该像素处理过
                p_bw[yy*624+(xx%1000)] = 255;
                if((xx/1000)==1)
dc.SetPixel((xx%1000)+11,512+10-yy,RGB(255,0,0));
                else if((xx/1000)==2)
dc.SetPixel((xx%1000)+11,512+10-yy,RGB(0,255,0));
                else if((xx/1000)==3)
dc.SetPixel((xx%1000)+11,512+10-yy,RGB(0,0,255));
                else
    dc.SetPixel((xx%1000)+11,512+10-yy,RGB(255,0,255));
                }
            }
            nStart++;
        }
        // 释放内存 */
        delete []GrowRegionx;
        delete []GrowRegiony;
    GrowRegionx = NULL ;
        GrowRegiony = NULL ;
}
```

遗传算法参数：染色体长度 20，种群大小为 20，交叉变异概率分别为 0.8 和 0.4，最大运行代数为 2000 代，程序运行结果如图 3 所示。

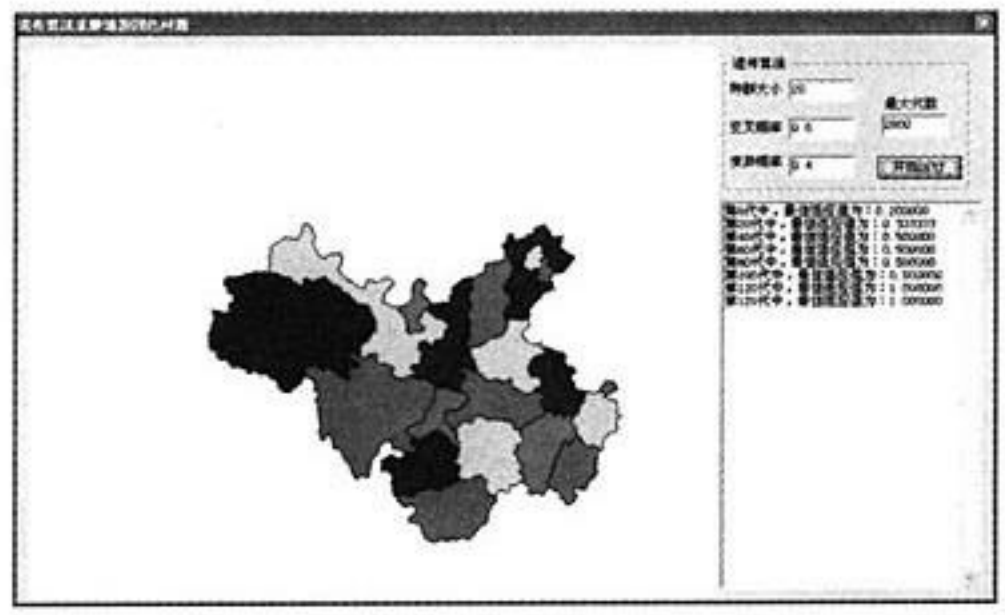


图 3　地图四色填充结果

4　结语

从程序的运行结果看，相比一般回溯算法和启发式搜索算法而言，遗传算法作为一种用于解决优化问题的近似算法，对于解决地图四色问题具有明显的优势：求解速度快，易于程序实现和算法移植，其算法的难点主要在于适应度函数的选取及算法参数的设置，不过这并不影响遗传算法成为求解大规模问题的有力工具之一。

（收稿日期：2012-11-12）

（上接第 70 页）

然后使用：

图 8　返回网页内容

```
for i:=0 to memo1.lines.count-2 do
begin
   memo2.Lines.add(copy (memo1.Lines [i],46-5,13) +'00');
// 分离出时间,放在收到指令时间框,
//为了按秒进行比对,在原来手机端上传的时间后面再加上 00
//表示秒钟
   memo3.Lines.add(copy (memo1.Lines [i],46+14-5,length
(memo1.Lines[i])-61+1) );
                //分离出指令内容,放到远程指令列表框
end ;
```

接下去只要用一个 timer 来判断，指令时间框中的当前行内容是否与系统时间一致，如果是，则去执行指令框中的当前行内容即可。

先定义：

```
command: = ' 外 部 程 序 \' + memo3.Lines [memo3.Lines.
Count-1 ] ;
  // 得到当前指令,外部程序文件夹中存放对应的批处理或者第
//三文件
```

然后进行时间比对，执行指令

```
label1.Caption  := FormatDateTime ('yyyymmdd-hhmmss', 
now);
if memo2.Lines [memo2.Lines.Count-1 ] =  label1.Caption
then  // 时间相等了就执行
begin
ShellExecute(0, nil, pchar(command), nil, nil, SW_SHOWNOR
MAL); //执行外部命令
Memo4.Lines.Add( memo3.Lines[memo3.Lines.Count-1 ] ) ;
// 把已执行代码加到已发指令中
End;
```

5　结语

系统在小米手机，安卓 V2.3.5 +Windows7 系统上实验通过。程序的更多功能期待有兴趣的读者进一步改进完善！

（收稿日期：2013-02-22）

电子琴程序开发

江 洪

摘 要：使用 VC6.0 开发了一个小巧的电子琴程序，可供练习弹奏电子琴使用，同时提供了演奏事先编辑好的乐谱文件的功能。音符采用事先录制好的声音文件，利用微软的多媒体库 winmm.lib 中的 API 进行声音播放，使用程序代码控制音符的实际演奏。

关键词：电子琴；演奏；音符；乐谱

电子琴是一种电子乐器，音量可以自由调节。音域较宽，表现力极其丰富。它可以模仿多种不同音色的乐器，甚至可以奏出常规乐器所无法发出的声音。电子琴适合于演奏节奏性较强的现代音乐，是电声乐队的中坚力量。

电子琴有一个类似钢琴的键盘，不同的键代表不同音高的音符。有规律地按下键盘，就可以演奏出旋律优美的乐曲。

下面就开发一个可以进行电子琴演奏的实用程序。程序使用电脑键盘模拟琴键。按下键盘就可出音符声，按键时间长短决定了音符发声的长短。同时程序还提供了演奏乐谱文件的功能，可以将一首乐谱按一定的规则放在一个文件中，由程序读取并播放。

1 相关知识

电子琴上的每个键代表某一个音高的音符。常用的音符有 7 个，分别唱作 do re mi fa so la xi。音符按照不同的音高，可分为中音部、高音部（提高 8 度）、重高音部（再提高 8 度）、低音部（降低 8 度）、重低音部（再降低 8 度）。音高是以半个音半个音地逐渐升高的。7 个音符中，除了 mi fa 之间和 xi do 之间间隔一个半音外，其他音符之间都间隔两个半音。乐谱中可以调整调号，以半个音为单位升高或者下降。

只有音符高低是不能构成乐曲的，还要有发声长短的区别。最长的音符称为全音符，然后是 2 分音符、4 分音符、8 分音符、16 分音符、32 分音符。一般 4 分音符称做一拍。一拍的长短是由乐谱的拍数决定的。如果乐谱的拍数为 120 拍/分钟，则一个全音符是 2 秒，一个 4 分音符就是 0.5 秒。每分钟拍数越多，乐谱越快，音符持续时间越短；反之，每分钟拍数越少，乐谱越慢，音符持续时间越长。

本程序提供了演奏乐谱的实用功能。可以事先将一首乐谱放在一个文件中，由程序进行读取并播放。

乐谱文件是文本文件。文件中有以下几类符号：

{+0}表示本乐谱的调号。大括号中的数字表示调号的升高或降低

[120]表示本乐谱的速度，以拍/分钟为单位

对于不同的音符，用不同的符号来表示

中音部 7 个音符，用 1 2 3 4 5 6 7 表示

高音部 7 个音符，用 A B C D E F G 表示

重高音部 7 个音符，用 H I J K L M N 表示

低音部 7 个音符，用 a b c d e f g 表示

重低音部 7 个音符，用 h i j k l m n 表示

休止符用 0 表示

每个音符后面都有一对小括号，里面数字表示该音符的长度，以 32 分音符为单位。如果音符中带有歌词，则在音符长度后面带有冒号，之后是歌词。

本程序所有音符均来源于事先录制好的声波 wav 文件。共录制了中音区、高音区，重高音区、低音区、重低音区共 60 个不同音高的音符。

程序界面上方显示当前行乐谱。

2 关键代码

在 OnPaint 函数中调用 DrawKeyBoard 画键盘。

在 PreTranslateMessage 函数中处理键盘按下和松开。按下键盘调用 PressKeyBoard 函数。松开键盘调用 ReleaseKeyBoard 函数。

准备音符缓冲区使用代码如下：

```
void PrepareSoundBuf()  //根据音调升降准备声音缓冲区
{
  int i,j,k[35],l,m;
  if(yf_regulation==0)
  {
      i=0;j=34;
      k[0]=0;k[1]=2;k[2]=4;k[3]=5;
      k[4]=7;k[5]=9;k[6]=11;k[7]=12;
      k[8]=14;k[9]=16;k[10]=17;k[11]=19;
      k[12]=21;k[13]=23;k[14]=24;k[15]=26;
      k[16]=28;k[17]=29;k[18]=31;k[19]=33;
      k[20]=35;k[21]=36;k[22]=38;k[23]=40;
      k[24]=41;k[25]=43;k[26]=45;k[27]=47;
```

```
    k[28]=48;k[29]=50;k[30]=52;k[31]=53;
    k[32]=55;k[33]=57;k[34]=59;
}
if(yf_regulation==1)
{
    i=0;j=33;
    k[0]=1;k[1]=3;k[2]=5;k[3]=6;
    k[4]=8;k[5]=10;k[6]=12;k[7]=13;
    k[8]=15;k[9]=17;k[10]=18;k[11]=20;
    k[12]=22;k[13]=24;k[14]=25;k[15]=27;
    k[16]=29;k[17]=30;k[18]=32;k[19]=34;
    k[20]=36;k[21]=37;k[22]=39;k[23]=41;
    k[24]=42;k[25]=44;k[26]=46;k[27]=48;
    k[28]=49;k[29]=51;k[30]=53;k[31]=54;
    k[32]=56;k[33]=58;
}
if(yf_regulation==2)
{
    i=0;j=33;
    k[0]=2;k[1]=4;k[2]=6;k[3]=7;
    k[4]=9;k[5]=11;k[6]=13;k[7]=14;
    k[8]=16;k[9]=18;k[10]=19;k[11]=21;
    k[12]=23;k[13]=25;k[14]=26;k[15]=28;
    k[16]=30;k[17]=31;k[18]=33;k[19]=35;
    k[20]=37;k[21]=38;k[22]=40;k[23]=42;
    k[24]=43;k[25]=45;k[26]=47;k[27]=49;
    k[28]=50;k[29]=52;k[30]=54;k[31]=55;
    k[32]=57;k[33]=59;
}
if(yf_regulation==3)
{
    i=0;j=32;
    k[0]=3;k[1]=5;k[2]=7;k[3]=8;
    k[4]=10;k[5]=12;k[6]=14;k[7]=15;
    k[8]=17;k[9]=19;k[10]=20;k[11]=22;
    k[12]=24;k[13]=26;k[14]=27;k[15]=29;
    k[16]=31;k[17]=32;k[18]=34;k[19]=36;
    k[20]=38;k[21]=39;k[22]=41;k[23]=43;
    k[24]=44;k[25]=46;k[26]=48;k[27]=50;
    k[28]=51;k[29]=53;k[30]=55;k[31]=56;
    k[32]=58;
}
if(yf_regulation==4)
{
    i=0;j=32;
    k[0]=4;k[1]=6;k[2]=8;k[3]=9;
    k[4]=11;k[5]=13;k[6]=15;k[7]=16;
    k[8]=18;k[9]=20;k[10]=21;k[11]=23;
    k[12]=25;k[13]=27;k[14]=28;k[15]=30;
    k[16]=32;k[17]=33;k[18]=35;k[19]=37;
    k[20]=39;k[21]=40;k[22]=42;k[23]=44;
    k[24]=45;k[25]=47;k[26]=49;k[27]=51;
    k[28]=52;k[29]=54;k[30]=56;k[31]=57;
    k[32]=59;
}
if(yf_regulation==5)
{
    i=0;j=31;
    k[0]=5;k[1]=7;k[2]=9;k[3]=10;
    k[4]=12;k[5]=14;k[6]=16;k[7]=17;
    k[8]=19;k[9]=21;k[10]=22;k[11]=24;
    k[12]=26;k[13]=28;k[14]=29;k[15]=31;
    k[16]=33;k[17]=34;k[18]=36;k[19]=38;
    k[20]=40;k[21]=41;k[22]=43;k[23]=45;
    k[24]=46;k[25]=48;k[26]=50;k[27]=52;
    k[28]=53;k[29]=55;k[30]=57;k[31]=58;
}
if(yf_regulation==6)
{
    i=0;j=30;
    k[0]=6;k[1]=8;k[2]=10;k[3]=11;
    k[4]=13;k[5]=15;k[6]=17;k[7]=18;
    k[8]=20;k[9]=22;k[10]=23;k[11]=25;
    k[12]=27;k[13]=29;k[14]=30;k[15]=32;
    k[16]=34;k[17]=35;k[18]=37;k[19]=39;
    k[20]=41;k[21]=42;k[22]=44;k[23]=46;
    k[24]=47;k[25]=49;k[26]=51;k[27]=53;
    k[28]=54;k[29]=56;k[30]=58;
}
if(yf_regulation==7)
{
    i=0;j=30;
    k[0]=7;k[1]=9;k[2]=11;k[3]=12;
    k[4]=14;k[5]=16;k[6]=18;k[7]=19;
    k[8]=21;k[9]=23;k[10]=24;k[11]=26;
    k[12]=28;k[13]=30;k[14]=31;k[15]=33;
    k[16]=35;k[17]=36;k[18]=38;k[19]=40;
    k[20]=42;k[21]=43;k[22]=45;k[23]=47;
    k[24]=48;k[25]=50;k[26]=52;k[27]=54;
    k[28]=55;k[29]=57;k[30]=59;
}
if(yf_regulation==-1)
{
    i=1;j=34;
    k[0]=1;k[1]=3;k[2]=4;k[3]=6;
    k[4]=8;k[5]=10;k[6]=11;k[7]=13;
    k[8]=15;k[9]=16;k[10]=18;k[11]=20;
    k[12]=22;k[13]=23;k[14]=25;k[15]=27;
    k[16]=28;k[17]=30;k[18]=32;k[19]=34;
    k[20]=35;k[21]=37;k[22]=39;k[23]=40;
    k[24]=42;k[25]=44;k[26]=46;k[27]=47;
    k[28]=49;k[29]=51;k[30]=52;k[31]=54;
    k[32]=56;k[33]=58;
}
```

```
if(yf_regulation==-2)
{
    i=2;j=34;
    k[0]=2;k[1]=3;k[2]=5;k[3]=7;
    k[4]=9;k[5]=10;k[6]=12;k[7]=14;
    k[8]=15;k[9]=17;k[10]=19;k[11]=21;
    k[12]=22;k[13]=24;k[14]=26;k[15]=27;
    k[16]=29;k[17]=31;k[18]=33;k[19]=34;
    k[20]=36;k[21]=38;k[22]=39;k[23]=41;
    k[24]=43;k[25]=45;k[26]=46;k[27]=48;
    k[28]=50;k[29]=51;k[30]=53;k[31]=55;
    k[32]=57;
}
if(yf_regulation==-3)
{
    i=3;j=34;
    k[0]=2;k[1]=4;k[2]=6;k[3]=8;
    k[4]=9;k[5]=11;k[6]=13;k[7]=14;
    k[8]=16;k[9]=18;k[10]=20;k[11]=21;
    k[12]=23;k[13]=25;k[14]=26;k[15]=28;
    k[16]=30;k[17]=32;k[18]=33;k[19]=35;
    k[20]=37;k[21]=38;k[22]=40;k[23]=42;
    k[24]=44;k[25]=45;k[26]=47;k[27]=49;
    k[28]=50;k[29]=52;k[30]=54;k[31]=56;
}
if(yf_regulation==-4)
{
    i=4;j=34;
    k[0]=3;k[1]=5;k[2]=7;k[3]=8;
    k[4]=10;k[5]=12;k[6]=13;k[7]=15;
    k[8]=17;k[9]=19;k[10]=20;k[11]=22;
    k[12]=24;k[13]=25;k[14]=27;k[15]=29;
    k[16]=31;k[17]=32;k[18]=34;k[19]=36;
    k[20]=37;k[21]=39;k[22]=41;k[23]=43;
    k[24]=44;k[25]=46;k[26]=48;k[27]=49;
    k[28]=51;k[29]=53;k[30]=55;
}
m=0;
for(l=i;l<=j;l++)
{
  memcpy(soundbuf[l],basebuf[k[m]],32000);m++;
}
}
```

播放音符使用代码：

```
//声音播放回调函数
static void CALLBACK waveOutCallback (HWAVEOUT
hwaveout,UINT uMsg,
                DWORD dwInstance,
                DWORD dwParam1,DWORD dwParam2)
{
  switch(uMsg)
  {
    case WOM_DONE:
        endflag=0;break;
    default:
      break;
  }
}
//播放音符 yf-待播放音符 speed-播放长度
void PlayYF(char yf,int speed)
{
  char buf1[500000],buf2[500000];
  HWAVEOUT hwo;
  WAVEHDR wavehdr;
  WAVEFORMATEX wfx;
  int i;
  if(endflag==1) return;
  //确定播放音符位置
  if(yf=='h') i=0;if(yf=='i') i=1;if(yf=='j') i=2;if(yf=='k') i=3;
  if(yf=='l') i=4;if(yf=='m') i=5;if(yf=='n') i=6;if(yf=='a') i=7;
  if(yf=='b') i=8;if(yf=='c') i=9;if(yf=='d') i=10;if(yf=='e') i=11;
if(yf=='f') i=12;if(yf=='g') i=13;if(yf=='1') i=14;if(yf=='2') i=15;
if(yf=='3') i=16;if(yf=='4') i=17;if(yf=='5') i=18;if(yf=='6') i=19;
if(yf=='7') i=20;if(yf=='A') i=21;if(yf=='B') i=22;if(yf=='C') i=23;
if(yf=='D') i=24;if(yf=='E') i=25;if(yf=='F') i=26;if(yf=='G') i=27;
if(yf=='H') i=28;if(yf=='I') i=29;if(yf=='J') i=30;if(yf=='K') i=31;
if(yf=='L') i=32;if(yf=='M') i=33;if(yf=='N') i=34;
//复制音符缓冲区
if(yf=='0')
  memset(buf1,0,32000);
else
  memcpy(buf1,soundbuf[i],32000);
memcpy(buf2,buf1,32000);
endflag=1;
//设置播放参数
wfx.cbSize=0;
wfx.nBlockAlign=2;
wfx.nAvgBytesPerSec=16000;
wfx.nChannels=1;
wfx.nSamplesPerSec=8000;
wfx.wBitsPerSample=16;
wfx.wFormatTag=1;
   waveOutOpen (&hwo, (UINT)WAVE_MAPPER,&wfx,
(DWORD)waveOutCallback,
            0,CALLBACK_FUNCTION);
 wavehdr.lpData=buf2;
 //设置播放缓冲区长度
   if (speed ==1) wavehdr.dwBufferLength =1000*120/
yp_speed*speed/1;
   if (speed ==2) wavehdr.dwBufferLength =2000*120/
yp_speed*speed/2;
 if(speed>2&&speed<=4)
wavehdr.dwBufferLength=4000*120/yp_speed*speed/4;
 if(speed>4&&speed<=8)
wavehdr.dwBufferLength=8000*120/yp_speed*speed/8;
```

GAME PROGRAM

```
  if(speed>8&&speed<=16)
wavehdr.dwBufferLength=16000*120/yp_speed*speed/16;
  if(speed>16)
wavehdr.dwBufferLength=32000*120/yp_speed*speed/32;
  wavehdr.dwBytesRecorded=0;
wavehdr.dwUser=0;wavehdr.dwFlags=0;wavehdr.dwLoops=0;
  waveOutPrepareHeader(hwo,&wavehdr,sizeof(WAVEHDR));
  waveOutWrite(hwo,&wavehdr,sizeof(WAVEHDR));  //播放
          waveOutUnprepareHeader   (hwo,&wavehdr,sizeof
(WAVEHDR));
  waveOutClose(hwo);
}
```

播放乐谱使用代码：

```
UINT PlayYP(LPVOID pParam)  //播放乐谱
{
  int i=0,j,k=0,key,speed;
  char s[100],s1[100],yf;
  CString str,str1;
  DWORD line=0;
  playing=1;
  str=yp;str.Replace(";","");
  str1="";
  while(1)
  {
      if(yp[k]==';')
      {
        line++;str1="";k++;
      }
      for(j=1;j<=100;j++) s[j-1]=str[i+j-1];
      for(j=1;j<=100;j++)
      {
        if(s[j-1]==')')
        {
       s[j]='\0';break;
        }
      }
      str1=str1+s;
      //复制一行乐谱
      strcpy(curstr,str1);
      if(strstr(curstr,".")!  =NULL) strcpy(curstr,"");
PostMessage(hwnd,WM_USER+10,0,0);  //显示当前行乐谱
      yf=s[0];
    for(j=1;j<=100;j++)
    {
      if(s[j-1]=='(')
      {
        strcpy(s1,s+j);break;
      }
    }
     for(j=1;j<=100;j++)
     {
  if(s1[j-1]==')'||s1[j-1]==':')
      {
       s1[j-1]='\0';break;
        }
      }
      speed=atoi(s1);
      //根据音符确定按键
    if(yf=='0') key=192;if(yf=='h') key=90;
    if(yf=='i') key=88;if(yf=='j') key=67;
    if(yf=='k') key=86;if(yf=='l') key=66;
    if(yf=='m') key=78;if(yf=='n') key=77;
    if(yf=='a') key=65;if(yf=='b') key=83;
    if(yf=='c') key=68;if(yf=='d') key=70;
    if(yf=='e') key=71;if(yf=='f') key=72;
    if(yf=='g') key=74;if(yf=='1') key=81;
    if(yf=='2') key=87;if(yf=='3') key=69;
    if(yf=='4') key=82;if(yf=='5') key=84;
    if(yf=='6') key=89;if(yf=='7') key=85;
    if(yf=='A') key=49;if(yf=='B') key=50;
    if(yf=='C') key=51;if(yf=='D') key=52;
    if(yf=='E') key=53;if(yf=='F') key=54;
    if(yf=='G') key=55;if(yf=='H') key=56;
    if(yf=='I') key=57;if(yf=='J') key=48;
    if(yf=='K') key=189;if(yf=='L') key=187;
    if(yf=='M') key=8;
      i=i+strlen(s);k=k+strlen(s);
    if(str[i-1]!  ='.')
      {
        PostMessage(hwnd,WM_KEYDOWN,key,0);  //按键盘
        PlayYF(yf,speed);  //播放音符
        Sleep(100);
        PostMessage(hwnd,WM_KEYUP,key,0);  //松开键盘
      }
      else
        break;
    while(1)
      {
      if(endflag==0) break;
        Sleep(1);
      }
  }
  playing=0;
  return 0;
}
```

3 结论

程序实现了基本的电子琴程序的弹奏和播放乐曲的功能。可以练习弹奏电子琴，也可编辑乐谱文件，欣赏乐曲的演奏。程序目录下已有一些乐谱文件可供演示使用，覆盖了儿童歌曲、影视歌曲、流行歌曲、外国歌曲、钢琴曲等不同的类型。读者也可按照规则制作新的乐谱文件并演奏。

（收稿日期：2012-10-12）

VC++开发垃圾清理软件

蔡智明 杨秋瑾

摘 要：分析了VC++程序设计开发的基本方法；详细说明了垃圾文件清理的原理和以绘制位图技术为背景的对话框绘制界面技术，重点介绍了垃圾清理程序的设计与实现。

关键词：垃圾清理；VC++程序设计；多线程；用户界面；位图；MFC

1 引言

垃圾文件是操作系统使用过一段时间生成的冗余文件，对操作系统来说是多余的，一般来说，每个一段时间应该给予清理，从而使系统更加干净，有助于节省磁盘空间，从而提高系统运行效率。下面主要介绍基于VC++开发的垃圾清理软件的设计与实现，包括功能的实现和界面的设计。

2 概述

垃圾文件清理，即对指定的文件格式的临时文件或垃圾文件进行遍历、扫描、显示、删除清理等。在程序界面设计方面，对默认对话框重新自定义绘制，主要包括标题栏的重绘、对话框边框的重绘、对话框背景重绘、以及最小化按钮、最大化按钮和关闭按钮等的重绘实现。经过界面的设计和功能的实现开发，从而开发出一款具有实用意义的垃圾清理工具。

3 编程平台与技术

实现的软件开发平台是基于Microsoft Visual Studio 2008集成开发环境，编程技术采用Visual C++编程技术，以及相关的开发软件如Photoshop CS5等。

4 需求分析

本程序的设计与开发主要分为两大模块，功能的设计开发和应用程序界面的设计开发。

功能的分析与设计：垃圾清理功能主要包括文件遍历扫描、显示已扫描到的文件以及垃圾文件的删除清理等。用户需要一边进行文件扫描，另一边可以对已经扫描到的垃圾文件进行清理操作。文件扫描通常会占用大量的时间，为了提高垃圾清理的可靠性和效率，应该使用多线程开发技术，即将文件扫描的任务放置在一个单独的线程中即可。

应用程序界面设计：在对话框重绘中，使用的主要技术有两个，一个是绘制对话框的背景位图，在对话框大小改变时能够输出位图，使位图能够适应对话框的大小。另一个是在对话框的指定区域输出位图。

5 程序开发与实现

5.1 垃圾文件的扫描、显示和清理

(1) 创建一个基于对话框的工程，工程名称为“ClearTmpFile”。

(2) 向对话框中添加静态文本框、按钮、组合框、列表框、进度条等控件，效果如图1所示。

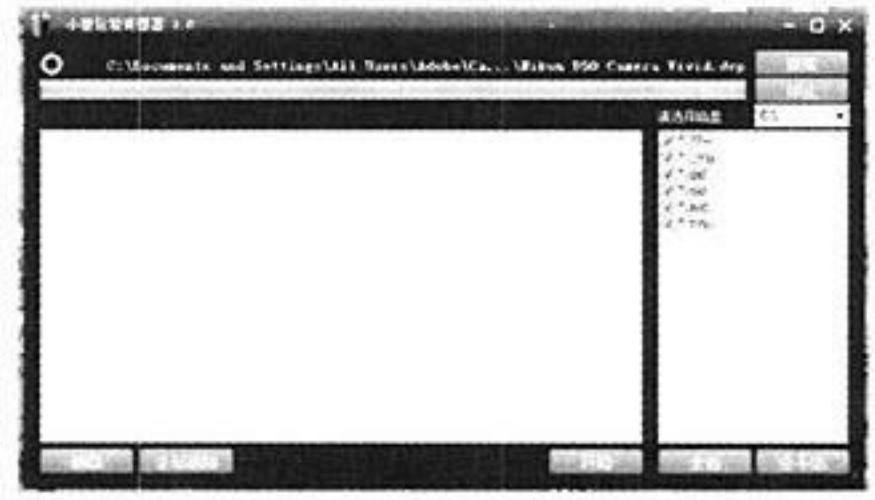


图1 控件布局界面图

(3) 在对话框类CClearTmpFileDlg中添加共有的主要数据成员，各成员功能见注释部分：

```
CList<CString, CString> m_fileExtList;//记录需要查找临
//时文件扩展名
bool m_bThreadExit;//线程是否退出
bool m_bFinding;//是否查找进行中
HANDLE m_hThread;//查找文件的线程句柄
HANDLE m_hThread2;//bmp 旋转线程句柄
CString m_szCurDisk;//查找的磁盘
HANDLE m_hEvent;//事件对象,在对话框关闭时将提前
//结束查找
bool m_bContinue;//判断暂停或继续按钮操作
DWORD GetDiskSize(char* strPath);//获取磁盘容量(已
//使用的)
DWORD m_dwDiskVol;//磁盘总容量大小,单位为 KB
DWORD m_dwScanedVol;//已扫描的文件的容量
DWORD m_dwScanedTmpFileVol;//扫描到的临时文件
//的容量大小
```

```
    DWORD m_dwScanedTmpFileNum;//扫描到的临时文件
//的容量大小
```

（4）向对话框类中添加 ResearchFile 方法，判读指定的目录，将指定的垃圾文件类型显示在扫描结果列表中：

```
void CClearTmpFileDlg::ResearchFile(char * pszPath)
{
    char szTmp[MAX_PATH]={0};//定义一个临时字符数组
    strcpy(szTmp,pszPath);
if(szTmp[strlen(szTmp)-1]! =\\')//将目录以"\\*.*"形式结尾
    {
        strcat(szTmp,"\\*.*");//连接字符串
    }
    else
    {
        strcat(szTmp,"*.*");//连接字符串
    }
WIN32_FIND_DATA findData;//定义一个文件查找数据结构
    memset(&findData,0,sizeof(WIN32_FIND_DATA));
    HANDLE hFind = FindFirstFile(szTmp,&findData);//开始
//查找文件
    //由于查找是在线程中进行的,这里判读用户是否退出线
//程,如果是则提前结束线程函数if(m_bThreadExit)
    {
        FindClose(hFind);//关闭查找句柄
        SetEvent(m_hEvent);//设置事件为有信号
        return;
    }
    if(hFind ! = INVALID_HANDLE_VALUE)//文件查找成功
    {
        while(FindNextFile(hFind,&findData)==TRUE)//查找
//下一个文件
        {//由于查找是在线程中进行的,这里判读用户是否退
//出线程,如果是则提前结束线程函数
            if (m_bThreadExit)
            {
                FindClose(hFind);//关闭查找句柄
        SetEvent(m_hEvent);//设置事件为有信号
                return;
            }
            //如果文件不是一个目录
                if (! (findData.dwFileAttributes &
FILE_ATTRIBUTE_DIRECTORY))
            {
                DWORD dwFileSize = (findData.
nFileSizeHigh* (MAXDWORD+1) + findData.nFileSizeLow)/
(1024);//获取文件大小,单位为 KB
                m_dwScanedVol += dwFileSize;//累计
//已扫描文件的容量大小,单位为 KB
                //m_dwScanedVol = m_dwScanedVol/
1024;//单位转换为:MB
                //设置进度条进度
                m_progressCtl.SetPos
((m_dwScanedVol/1024));
                char szFileName[MAX_PATH] = {0};
//定义字符数组,存储完整的文件名
            strcpy(szFileName,pszPath);//获取完整文件名
            if(szFileName[strlen(szFileName)-1] ! = \\')
                {
                    strcat(szFileName,"\\");
                }
            strcat(szFileName,(char *)findData.cFileName);
                    if(IsTmpFile(szFileName))//判断
//szFileName 是否是临时文件
                    {
                m_dwScanedTmpFileVol += dwFileSize;
//累计扫描到的临时文件容量大小,单位为 KB
                    m_dwScanedTmpFileNum ++;
//累计扫描到的临时文件的数目
                    m_listBoxResults.AddString
((LPCTSTR)szFileName);
                    }
                }
                else//如果文件是一个目录,则递归遍历该目录
                {
if((strcmp((const char *)&findData.cFileName,"...")! =0) &&
    (strcmp((const char *)&findData.cFileName,"..")! =0)&&
    (strcmp((const char *)&findData.cFileName,".")! =0))
                {
                    char szFileName[MAX_PATH]={0};
            strcpy(szFileName,pszPath);//获取完整文件名
                if(szFileName[strlen(szFileName)-1]! =\\')
                    {
                        strcat(szFileName,"\\");
                    }
            strcat(szFileName,(char *)findData.cFileName);
                    //由于查找是在线程中进行的,这里
//判读用户是否退出线程,如果是则提前结束线程函数
                    if(m_bThreadExit)
                    {
                FindClose(hFind);//关闭查找句柄
            SetEvent(m_hEvent);//设置事件为有信号
                    return;
                    }
                    ResearchFile(szFileName);//递归调用
                }
            }
        }
    }
    FindClose(hFind);//关闭文件查找句柄
}
```

（5）定义线程函数，用来单独执行扫描查找垃圾文件任务：

```
DWORD _stdcall FindTmpFile(LPVOID lpParameter)
{
```

```
        CClearTmpFileDlg* pDlg = (CClearTmpFileDlg*)
lpParameter;//获取线程参数
    WaitForSingleObject(pDlg->m_hEvent,INFINITE);//等待
//事件有信号
    CString dir = pDlg->m_szCurDisk.GetBuffer();//根据当
//前盘符目录磁盘目录
    char *s = (LPSTR)(LPCTSTR)dir;
    pDlg->ResearchFile(s);
    pDlg->Restore();
  pDlg->ShowResultText();//显示扫描临时文件的数目和大小
    //恢复数据为初始状态
    pDlg->m_dwScanedTmpFileVol = 0;
    pDlg->m_dwScanedTmpFileNum = 0;
    pDlg->m_fileExtList.RemoveAll();
    return 0;
}
```

(6) 处理“立即扫描”或“开始”按钮的单击事件，创建一个新的线程执行扫描文件的任务：

```
//如果查找没有结束,则不允许开始新的文件查找
    GetDlgItem(IDC_BEGIN)->ShowWindow(SW_HIDE);
    if(! m_bFinding && GetTmpExtName())//获取文件扩展名
    {
        GetDlgItem  (IDC_PROGRESS1) ->ShowWindow
(TRUE);//显示进度条
    GetDlgItem(IDC_LIST1)->ShowWindow(SW_SHOW);
        m_bThreadExit  = FALSE;
        m_bFinding = TRUE;
  m_combox.GetWindowText(m_szCurDisk);//获取当前盘符
        // 初始化进度条相关数据
        m_dwDiskVol  =GetDiskSize  ((LPSTR) (LPCTSTR)
m_szCurDisk);//获取当前磁盘的容量大小(已使用的)
        CString str;
        str.Format(_T("%d"),m_dwDiskVol/(1024));
        double iSize = atoi(str);
        m_progressCtl.SetRange32 (0,m_dwDiskVol/1024);
//初始化进度条,设置进度条的范围,范围为 MB 的数量
        if(m_hEvent! =NULL)
        {
            CloseHandle(m_hEvent);//关闭事件对象
            m_hEvent = NULL;
        }
    m_listBoxResults.ResetContent();//清空查找结果列表
        m_hEvent = CreateEvent  (NULL,FALSE,TRUE,_T("
Event"));//创建事件对象
        //创建一个线程,开始执行线程函数
m_hThread = CreateThread(NULL,0,FindTmpFile,this,0,NULL);
m_hThread2 = CreateThread(NULL,0,RotatingImg,this,0,NULL);
    UpdateData(FALSE);
```

(7) 清理已扫描到的垃圾文件，即采用删除文件策略，使用 DeleteFile () 方法：

```
void CClearTmpFileDlg::OnBnClickedDelall()
{//删除已扫描到的垃圾文件
    CString strDel;
    CFile file;
    for(int i=0;i<m_listBoxResults.GetCount();i ++)
    {
        m_listBoxResults.GetText(i,strDel);
        GetDlgItem(IDC_TEST)->SetWindowText(strDel);
        DeleteFile(strDel);//删除指定路径的文件
    }
    m_listBoxResults.ResetContent();
 GetDlgItem(IDC_TEST)->SetWindowText(_T("清理完毕！"));
    GetDlgItem(IDC_LIST1)->ShowWindow(SW_HIDE);
    GetDlgItem(IDC_BEGIN)->ShowWindow(SW_SHOW);
GetDlgItem(IDC_BEGIN)->SetWindowText(_T("重新扫描"));
}
```

到此，软件的主要功能部分都已经开发完毕，下面将进行对软件界面的设计与开发。

5.2 程序界面的设计与实现

应用程序界面的设计包括两部分，一部分是对话框自身的重设计，二是对话框控件的重绘，本程序主要对按钮控件进行重绘设计。

5.2.1 绘制对话框的背景位图

绘制对话框背景位图本文采用的是处理对话框的 WM_PAINT 消息，该消息初始化时候对对话框进行绘制，从而绘制背景位图。绘制背景位图的主要代码如下：

```
    CRect  rect;
    CPaintDC  dc(this);
    GetClientRect(&rect); //获取客户区
//设置对话框背景颜色
    dc.FillSolidRect(rect,RGB(14,94,157));  //设置为窗口背景
```

5.2.2 在指定的区域中输出位图

为了能够在指定的区域中输出位图，需要使用设备上下文 CDC 类的 StretchBlt 方法。由于我们需要在窗口的非客户区域绘制位图，因此需要使用 CWindowDC 类的 StretchBlt 方法，CWindowDC 类派生与 CDC 类，它提供了在窗口非客户区域绘制位图的功能。该方法数从源矩形中复制一个位图到目标矩形，必要时按目前目标设备设置的模式进行图像的拉伸或压缩。输出位图的主要实现代码如下：

```
    CRect winRC;
    CDC* pDC=GetWindowDC();//获取窗口设备上下文
    CDC memDC;
    memDC.CreateCompatibleDC(pDC);//创建兼容内存位图
    BITMAPINFO bmpInfo;
    CBitmap bmp;  //定义位图对象
    GetWindowRect(&winRC);
    bmp.LoadBitmap(nID);//加载位图
      bmp.GetObject(sizeof(BITMAPINFO),&bmpInfo);//获取
//位图信息
```

```
    int nBmpCX = bmpInfo.bmiHeader.biWidth;//获取位图
//宽度
    int nBmpCY = bmpInfo.bmiHeader.biHeight;//获取位图
//高度
    memDC.SelectObject(bmp);//选中位图对象
    pDC->StretchBlt(x,y,w,h,
        &memDC,0,0,nBmpCX,nBmpCY,SRCCOPY);//在窗
//口中绘制位图
    bmp.DeleteObject();//释放位图对象
    ReleaseDC(pDC);//释放 DC
}
```

5.2.3 对话框绘制的实现

在对话框重绘的设计与实现过程中，一般需要绘制的对话框区域主要有标题部分、边框部分和客户区部分。具体的区域划分如图 2 所示。

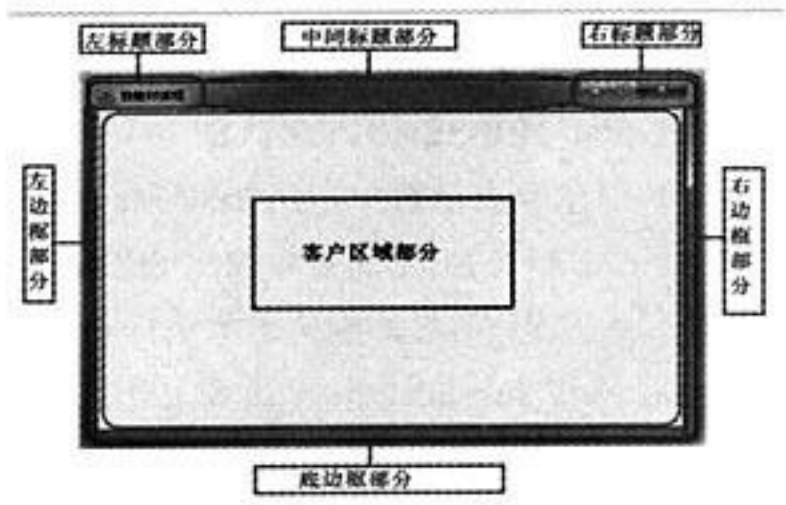


图 2　对话框绘制区域图

既然要对多个区域进行位图显示输出，所以我们先封装一个 bmp 位图显示输出函数如下：

```
void CClearTmpFileDlg::DisplayBmp (int x,int y,int w,int h,int
nID)
{//nID 表示位图资源的 ID
    CRect winRC;
    CDC* pDC=GetWindowDC();
    CDC memDC;
    memDC.CreateCompatibleDC(pDC);
    BITMAPINFO bmpInfo;
    CBitmap bmp;
    GetWindowRect(&winRC);
    bmp.LoadBitmap(nID);
    bmp.GetObject(sizeof(BITMAPINFO),&bmpInfo);
    int nBmpCX = bmpInfo.bmiHeader.biWidth;
    int nBmpCY = bmpInfo.bmiHeader.biHeight;
    memDC.SelectObject(bmp);
    pDC->StretchBlt(x,y,w,h,
        &memDC,0,0,nBmpCX,nBmpCY,SRCCOPY);//在窗
//口中绘制位图
    bmp.DeleteObject();
    ReleaseDC(pDC);
}
```

对各个区域进行位图输出重绘。由于标题栏以及边框主要都是非客户区域绘制，因此应该在 WM_NCPAINT 消息中绘制。当然得先通过添加资源的方式将所用到的 bmp 位图资源导入到项目中。

在 WM_NCPAINT 消息对于的 方法 OnNcPaint（）中调用对话框绘制方法 DrawDialog（）。该方法的功能就是绘制对话框各个区域的位图。主要代码如下：

```
void CClearTmpFileDlg::DrawDialog()
 {//重绘对话框标题栏、边框、最小化按钮、最大化按钮和关闭按
//钮等界面
                m_nFrameCY = GetSystemMetrics
(SM_CYFIXEDFRAME);//获取对话框边框的高度
                m_nFrameCX = GetSystemMetrics
(SM_CXDLGFRAME);//获取对话边框的宽度
    if(GetStyle()&WS_BORDER)//获取对话框是否有边框
    {
                m_nBorderCY = GetSystemMetrics
(SM_CYBORDER) + m_nFrameCY;
                m_nBorderCX = GetSystemMetrics
(SM_CXBORDER) +m_nFrameCX;
    }
    else
    {
        m_nBorderCY = m_nFrameCY;
        m_nBorderCX = m_nFrameCX;
    }
    m_nTitleBarCY = GetSystemMetrics (SM_CYCAPTION)
+ m_nBorderCY;//计算标题栏高度
    m_nTitleBarCX =m_nBorderCX;
    CRect winRect,factRect;
    GetWindowRect(&winRect);
    factRect.CopyRect (CRect (0,0,winRect.Width (),winRect.
Height()));
    CWindowDC windowsDC(this);//获取窗口设备上下文
    //获取整个 MFC 窗口的高度和宽度
    m_nWinWidth = winRect.Width();//=781
    m_nWinHeight = winRect.Height();//=459
    //绘制对话框左标题栏位图
    DisplayBmp(0,0,100,m_nTitleBarCY,IDB_LEFTTITLE);
    //绘制对话框标题栏左端的 logo 图标
    DisplayBmp(3,0,26,m_nTitleBarCY,IDB_APPICON);
    //绘制对话框右标题栏位图
    DisplayBmp  (m_nWinWidth -100,0,100,m_nTitleBarCY,
IDB_RIGHTTITLE);
    //绘制对话框中标题栏位图
    DisplayBmp  (100,0,m_nWinWidth -200,m_nTitleBarCY,
IDB_MIDTITLE);
    //绘制对话框左边框位图
            DisplayBmp  (0,m_nTitleBarCY,m_nBorderCX,
m_nWinHeight-m_nBorderCY,IDB_LEFTBAR);
    //绘制对话框底边框位图
            DisplayBmp  (m_nBorderCX,m_nWinHeight -
m_nBorderCX,m_nWinWidth-2*m_nBorderCX,m_nBorderCX,
```

```
IDB_BOTTOMBAR);
    //绘制对话框左边框位图
            DisplayBmp  (m_nWinWidth -m_nBorderCX,
m_nTitleBarCY,m_nBorderCX,m_nWinHeight -m_nBorderCY,
IDB_RIGHTBAR);
    //给对话框绘制最小化按钮
DisplayBmp(m_nWinWidth-26*3-5,0,26,26,IDB_MINBTN1);
    //给对话框绘制最大化按钮
DisplayBmp(m_nWinWidth-26*2-5,0,26,26,IDB_MAXBTN1);
    //给对话框绘制关闭按钮
DisplayBmp(m_nWinWidth-26-5,0,26,26,IDB_CLOSEBTN1);
    ReleaseDC(&windowsDC);
    //ReleaseDC(&memDC);
    DrawTitleBarText();//输出标题栏文本
}
```

上面代码中最后的绘制对话框标题文本的方法 DrawTitleBarText () 的主要代码如下：

```
CString strTitle ="小蔡垃圾清理器 3.0";
CDC* pDC= GetWindowDC();
pDC->SetBkMode(TRANSPARENT);
pDC->SetTextColor(RGB(255,255,255));
pDC->SetTextAlign(TA_CENTER);
CRect rect;
GetClientRect(&rect);
CSize szText = pDC->GetTextExtent(strTitle);
CFont* font,*fOldFont;
font = new CFont;
font    ->CreateFont    (12,0,0,0,FW_BOLD,FALSE,FALSE,0,
ANSI_CHARSET , OUT_DEFAULT_PRECIS , CLIP_DEFAULT_
PRECIS,DEFAULT_QUALITY,FF_SWISS,_T("宋体"));
fOldFont = pDC->SelectObject(font);
pDC->TextOut(100,6.5,strTitle,18);
pDC->SelectObject(fOldFont);
ReleaseDC(pDC);
```

在完成对话框相应区域的位图后，并没有完成任务，还需要处理标题栏按钮的热点效果，以及按钮的单击事件。首先得处理鼠标在非客户区域移动时的事件，即 WM_NCMOUSEMOVE 消息，在其消息处理函数中判断当前的鼠标点是否位于标题栏的按钮区域，如果是则设置按钮的热点效果，并且记录当前的按钮状态，及鼠标点在哪个按钮上。同样，处理对话框非客户区域的单击事件，即 WM_NCLBUTTONDOWN 消息，在其消息处理函数中完成单击事件操作。这部分的代码比较简单，在此不予显示。

5.2.4 按钮控件重绘的实现

在 MFC 下编程，很多时候对于标准的按钮控件不是很满意，想要弄得美观些。这就需要按钮重绘。重绘按钮一般的实现方法就是重写 CButton 类。

首先给工程添加一个自绘按钮类 MyDrawButton，基类为 CButton。要想让按钮具备自绘功能，就要为按钮添加 BS_OWNERDRAW 属性。为类 CButton 重载 PreSubclassWindow 虚函数。在该函数中添加如下一行代码：

```
SetButtonStyle(GetButtonStyle() | BS_OWNERDRAW);
```

当按钮控件具有了自绘功能之后，每次控件状态改变都会触发 DrawItem 函数，在该函数中来绘制按钮的形态外观，所以第二步就要重载 DrawItem 虚函数。在这个函数中就可以自由发挥了，比如绘制背景、底色、按钮标题、绘制文本字体样式等。

一般都会为按钮定义几种不同状态时的外观，比如光标滑过时的状态，按钮按下时的状态，按钮禁用时的状态，以及按钮的正常状态等。这就要为新的按钮添加几种重要的消息响应。比如 WM_MOUSELEAVE 消息，WM_MOUSEHOVER 消息和 WM_MOUSEMOVE 消息等，值得一提的是前两个消息的响应函数需要自己手动添加，微软提供了一个 TrackMouseEvent 函数在光标离开一个窗口时投递 WM_MOUSELEAVE 消息，光标滑过窗口时投递 WM_MOUSEHOVER 消息。一般来说可以在 WM_MOUSEMOVE 消息响应函数中调用 TrackMouseEvent 函数来投递 WM_MOUSELEAVE 消息和 WM_MOUSEHOVER 消息。然后在 WM_MOUSELEAVE 消息的响应函数中标记“光标已经离开按钮”，然后调用 InvalidateRect 函数让按钮重绘。在 WM_MOUSEHOVER 消息的响应函数中标记“光标正在按钮上方”，并调用 InvalidateRect 函数让按钮重绘。

（1）绘制按钮背景样式，即绘制背景 bmp 位图，使得按钮具有自定义的样式，同时在绘制按钮背景的输出位图时采用 TransparentBlt（）函数，该函数的作用是使窗体上显示位图的背景与窗体背景色融为一体，不仅可以显示按钮 bmp 位图样式，而且还可以使背景透明。

（2）就是绘制按钮上的文本。主要绘制按钮上文本的样式，包括字体大小，字体样式，字体颜色等属性。

（3）实现不同状态下的按钮的外观样式，主要包括 WM_MOUSEMOVE 和 WM_MOUSELEAVE 两个消息的消息处理函数。分别实现鼠标在按钮区域上和不在按钮区域上的状态。为了标记鼠标移动到按钮区域内停留，需要用到一个定时器来标记鼠标是否还在按钮区域内停留。在 WM_MOUSEMOVE 内启动定时器，触发 WM_MOUSELEAVE 消息时结束定时器即销毁定时器。定时器的主要代码如下：

```
void MyDrawButton::OnTimer(UINT_PTR nIDEvent)
{
    // TODO: 在此添加消息处理程序代码和/或调用默认值
    if(nIDEvent ! = 24)
        return;
    CPoint      point;
    CRect       rect;
    GetWindowRect(&rect);
    GetCursorPos(&point);
    // 如果鼠标离开按钮区域，重绘按钮
```

```
if (! rect.PtInRect(point) && m_bMove)
{
        KillTimer (24);
        m_DrawState=ST_MOVEOUT;
        m_bMove=FALSE;
        Draw();
}
CButton::OnTimer(nIDEvent);
}
```

重绘按钮类 MyDrawButton 的主要实现代码如下：

定义的一些重绘用到的变量：

```
 #define ST_MOVEIN   0//绘制状态—在按钮区域上
#define ST_MOVEOUT  1 //绘制状态—不在按钮区域上
int m_DrawState;//绘制状态
 int m_nBmpID;//当前显示的背景 bmp 位图的资源 ID
 bool m_bMove;//鼠标是否进入按钮区域
 COLORREF m_clText;//当前文本颜色
 COLORREF m_clActiveText;//鼠标进入按钮区域时文本
//颜色
 COLORREF m_clNormalText;//鼠标离开按钮区域时文本
//颜色
```

消息处理函数和定义的函数：

```
void MyDrawButton::PreSubclassWindow()
{
    SetButtonStyle(GetButtonStyle() | BS_OWNERDRAW);
}
void    MyDrawButton::DrawItem    (LPDRAWITEMSTRUCT
lpDrawItemStruct)
{
    Draw();//绘制按钮
}
void MyDrawButton::Draw()//绘制按钮
{
DrawBackground();//绘制按钮 bmp 位图,并使背景透明化
    DrawText();//绘制按钮上的文本
}
void MyDrawButton::DrawText()
{//绘制按钮上的文本的字体大小、样式等
    CString          itemString;
    CRect  clientRect;
    CClientDC  dc(this);
    GetClientRect(&clientRect);
    GetWindowText(itemString);
    if(itemString)
    {
        CSize size=dc.GetTextExtent (itemString);//获得所
//选字体中指定字符串的高度和宽度
        int rectwidth=clientRect.Width();
        int rectheight=clientRect.Height();
        int textwidth=size.cx ;
        int textheight=size.cy ;
int x,y; // 文本的位置
// 计算文本的输出位置
x=(rectwidth-textwidth)/2;//水平居中
y=(rectheight-textheight)/2;//垂直居中
switch(m_DrawState)
{
case ST_MOVEIN://鼠标进入按钮区域
        m_clText=m_clActiveText;
        break;
        case ST_MOVEOUT://鼠标离开按钮区域
            m_clText=m_clNormalText;
            break;
        default:
            m_clText=m_clNormalText;
            break;
        }
        dc.SetTextColor(m_clText);
        dc.SetBkMode(TRANSPARENT);
        CFont *font ;
        font =new CFont();
            int fontSize = 14;font  ->CreateFont  (fontSize,
0,0,0,FW_BOLD,FALSE,FALSE,0,ANSI_CHARSET,
OUT_DEFAULT_PRECIS,CLIP_DEFAULT_PRECIS,
DEFAULT_QUALITY,FF_SWISS,_T("宋体"));
        dc.SelectObject(font);
        dc.TextOut (x,y,itemString);
    }
}
void MyDrawButton::SetBkBmp(int nBmpID)
{//设置按钮 bmp 位图样式
    m_nBmpID = nBmpID;
}
void MyDrawButton::DrawBackground()
{//绘制按钮 bmp 位图,并使背景透明化
    CRect winRC;
    CDC* pDC=GetWindowDC();
    CDC memDC;
    memDC.CreateCompatibleDC(pDC);
    BITMAPINFO bmpInfo;
    CBitmap bmp;
    GetWindowRect(&winRC);
    bmp.LoadBitmap(m_nBmpID);
    bmp.GetObject(sizeof(BITMAPINFO),&bmpInfo);
    int nBmpCX = bmpInfo.bmiHeader.biWidth;
    int nBmpCY = bmpInfo.bmiHeader.biHeight;
    memDC.SelectObject(bmp);
pDC->TransparentBlt(0,0,nBmpCX,nBmpCY,&memDC,0,0,
        nBmpCX,nBmpCY,RGB(14,94,157));//在窗口中绘制
//位图,RGB(14,94,157)是透明色
    bmp.DeleteObject();
    ReleaseDC(pDC);
}
```

到此，按钮的自定义重绘完成了，接下来就可以使用自己重绘的按钮类 MyDrawButton 了。首先往对话框中添加一个按钮控件（以立即扫描按钮为例），假设它的 ID 值为 IDC_TEST。进入类向导（Class Wizard）的成员变量属性页，为 IDC_BEGIN 添加一个变量 m_btnBegin。如下：

```
MyDrawButton m_btnBegin;
```

然后就可以调用 MyDrawButton 的方法来设置按钮的样式了。如下：

```
m_btnBegin.SetBkBmp(IDB_BTN210x95,IDB_BTN210x95_3);
//IDB_BTN210x95,IDB_BTN210x95 分别为默认位图和鼠标在
//按钮区域时的位图。
```

到现在为止，按钮类的重绘完成了，可以随意定义自己喜欢的样式的按钮了。现在相对完善成形的一个垃圾清理工具软件就开发完了。

最后软件的运行效果界面如图 3-图 6 所示。

图 3 软件打开准备就绪界面

图 4 正在运行扫描界面

图 5 扫描完成界面

图 6 清理完成界面

6 结语

通过 VC++软件开发的需求分析、功能分析设计、软件功能的编程实现，了解了 VC++应用程序尤其是 MFC 应用系统设计与开发的流程和解决方案；掌握了 VC++编程技术和面向对象技术以及 bug 的调试技术和解决 bug 的能力，重点掌握了文件的查找、遍历、循环以及删除等和应用程序界面设计等知识和技术，同时也学会了如何设计并实现 VC++应用程序主界面的设计与美化。在设计界面过程中，渐渐地学会了如何设计漂亮、美观、友好的用户界面；通过这次设计与开发，使自己懂得如何在困难重重中一步一步细心地发现问题，解决问题。另外，提高了对编程认知与总结，不断加强编程基本功，不断总结经验，学习他人的优秀成果，并提高了独立思考和解决问题的能力。

（收稿日期：2013-02-18）

（上接第 72 页）

果要好。

（2）该算法实现目标的提取与种子点的选取和给定的阈值有关。

（3）该方法对图像分割的研究有一定的借鉴和指导作用。

参考文献

[1] 范立南，韩晓微，张广渊. 图像处理与模式识别 [M]. 北京：科学出版社，2007：26-30.

[2] 姚敏，等. 数字图像处理 [M]. 浙江：浙江大学出版社，2008.

[3] 韩晓微，杨喆，李彦平，等. 基于颜色相似系数的彩色图像分割方法 [J]. 沈阳大学学报，2004，16（6）：14-17.

[4] 杨旭强，冯勇，刘洪臣. 一种基于 HSI 颜色模型的目标提取方法 [J]. 光学技术，2006，32（2）：290-292.

[5] 王晓飞，郭敏，徐秋平. 基于图割与改进水平集的目标提取方法. 计算机工程 [J]，2010，36（22）：214-216.

（收稿日期：2012-11-30）

通过驱动程序实现开机自动运行程序

刘惠宁 屈剑平

摘 要：利用 Windows 启动过程中启动服务程序，在服务程序（通过驱动程序实现）中写注册表相关项，自动运行程序。

关键词：服务；驱动程序；DDK 环境；注册表

1 引言

目前开机自动运行程序的方法很多，有利用 INI 文件的，也有利用注册表的，但这些方法都有一个明显的缺点，就是太直接了。现有的许多杀毒、安全软件很容易清除掉，隐蔽性很差。

2 原理

2.1 Windows 服务

Windows 启动过程中，会启动注册表中 [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services] 项下的服务程序。每个服务程序在该项下有一个“子键”，“子键”就是服务名，即用“net start 服务名”命令启动服务时的“服务名”。譬如“假脱机”服务有[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Spooler] 项，其中“Spooler”就是子键。该子键下一般有“Type”、“Start”、“ErrorControl”、“DisplayName”、“Group”和“ImagePath”键名。其中：

（1）“Type”表示驱动程序的种类，1 表示设备驱动程序。

（2）“Start”表示驱动程序的起始启动时间，2 表示在 SCM 进程（Services.exe）启动以后 SCM 启动该驱动程序或服务。

（3）“ErrorControl”表示驱动装入失败的错误处理，1 表示如果驱动程序或者服务报告了一个错误，则显示一个警告。

（4）“DisplayName”表示服务的显示名称。显示在“服务”管理器中，譬如“假脱机”服务的显示名称是“Print Spooler”。

（5）“Group”表示组的名称。

（6）“ImagePath”表示服务或驱动程序可执行文件的路径。

2.2 Windows 驱动程序

Windows 驱动程序的入口函数是 DriverEntry，只要编程该函数在其中写 2.1 步所要求的注册表项即可。

2.3 自动启动程序

由 2.2 步的驱动程序决定自动启动程序的位置和名称。

3 驱动程序编辑、编译和链接步骤

（1）在 D 盘根目录下建立目录 autorun。

（2）在 autorun 目录下用记事本程序或其他纯文本程序建立 Driver.h 文件，输入以下内容：

```
/*********************************************************
* 文件名称:Driver.h
* 作    者:刘惠宁 屈剑平
* 完成日期:2012-12-29
*********************************************************/
#pragma once
#ifdef __cplusplus
extern "C"
{
#endif
#include <NTDDK.h>
#ifdef __cplusplus
}
#endif
#define PAGEDCODE code_seg("PAGE")
#define LOCKEDCODE code_seg()
#define INITCODE code_seg("INIT")
#define PAGEDDATA data_seg("PAGE")
#define LOCKEDDATA data_seg()
#define INITDATA data_seg("INIT")
```

（3）在 autorun 目录下用记事本或其他纯文本程序建立 Driver.cpp 文件，输入以下内容：

```
/*********************************************************
* 文件名称:Driver.cpp
* 作    者:刘惠宁 屈剑平
* 完成日期:2012-12-29
*********************************************************/
#include "Driver.h"
/*********************************************************
* 函数名称:DriverEntry
* 功能描述:初始化驱动程序,定位和申请硬件资源,创建内核对象
* 参数列表:
    pDriverObject:从 I/O 管理器中传进来的驱动对象
```

```
    pRegistryPath:驱动程序在注册表的中的路径
* 返回值:返回初始化驱动状态
**************************************************/
#pragma INITCODE
extern "C" NTSTATUS DriverEntry (
        IN PDRIVER_OBJECT pDriverObject,
        IN PUNICODE_STRING pRegistryPath)
{
   NTSTATUS status, ret;
   KdPrint(("Enter DriverEntry\n"));
     HANDLE hReg;
     OBJECT_ATTRIBUTES obj;
     UNICODE_STRING keyname;
                    RtlInitUnicodeString (&keyname, L"
\\REGISTRY\\MACHINE\\SOFTWARE\\Microsoft\\Win-
dows\\CurrentVersion\\Run");
     InitializeObjectAttributes(&obj,
          &keyname,
          0,
          NULL,
          NULL);
     NTSTATUS RetVal = ZwOpenKey(&hReg,
          KEY_ALL_ACCESS,
          &obj);
     UNICODE_STRING runx;
     RtlInitUnicodeString(&runx, L"XRW");
                    WCHAR MyData [500] = L"C:
\\WINDOWS\\system32\\xrwrun.exe";
     KdPrint(("MyData is : %wS\n", MyData));
     ret = ZwSetValueKey (hReg, &runx, 0, REG_SZ,
MyData, wcslen(MyData)*2 + 2);
     ZwClose(hReg);
     KdPrint(("DriverEntry end\n"));
   status = STATUS_SUCCESS;
     return status;
}
```

其中驱动程序中的“\\REGISTRY\\MACHINE\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run”相当于注册表中的“HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services”。

（4）在 AutoRun 目录下用记事本或其他纯文本程序生成 Sources 文件，内容如下：

```
TARGETNAME=AutoRun
TARGETTYPE=DRIVER
TARGETPATH=OBJ
INCLUDES=$(BASEDIR)\inc;\
      $(BASEDIR)\inc\ddk;\
SOURCES=Driver.cpp\
```

（5）在 AutoRun 目录下用记事本或其他纯文本程序生成 makefile 文件，内容如下：

```
#
# DO NOT EDIT THIS FILE!!!  Edit .\sources. If you want to
add a new source
# file to this component.  This file merely indirects to the real
make file
# that is shared by all the driver components of the
Windows NT DDK
#
! INCLUDE $(NTMAKEENV)\makefile.def
```

（6）用 DDK 环境生成 AutoRun.sys 文件。

在 Windows 的开始菜单中选择“Windows XP Free Build Environment”或“Windows XP Checked Build Environment”编译环境后，进入命令行窗口，输入以下命令进入需要编译的目录 d:

```
cd \Autorun
```

再输入“Build”即可在“D:\AutoRun\objfre_wxp_x86\i386”或“D:\AutoRun\objchk_wxp_x86\i386”目录下生成“AutoRun.sys”文件。

4 生成服务环境

（1）用记事本或其他纯文本程序编写“AutoRun.reg”文件，保存位置不限。输入以下内容：

```
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Ser-
vices\AutoRun]
"Type"=dword:00000001
"Start"=dword:00000002
"ErrorControl"=dword:00000001
"Group"="TDI"
"DisplayName"="XRW Driver"
"ImagePath"="System32\\drivers\\AutoRun.sys"
```

（2）双击“AutoRun.reg”文件，导入到注册表中。

（3）将之前生成的“AutoRunsys”文件复制到“C:\Windows\system32\driver”目录下，由上步中的 ImagePath 确定。

（4）将需要自动运行的文件改名为“xrwrun.exe”，复制到“C:\Windows\system32”目录下，该文件的位置和名称由驱动程序中的“WCHAR MyData [500] = L" C:\\WINDOWS\\system32\\xrwrun.exe" ;”语句决定。

（5）重启计算机，就可看到自动运行了需要的文件。

5 编程及测试环境

（1）编程环境：Visual C++ 6.0。

（2）DDK 环境：Windows XP 3790.1218。

（3）运行环境：Windwos 2003/XP。

6 调试及显示问题

由于该驱动程序没有卸载函数，所以导致用“DriverMon

（下转第 93 页）

TROUBLESHOOTING OF PROGRAM

Q 如何在外部获取 IE 文档对象

A 分析网页内容时，通常可以在自己的应用程序里内嵌 WebBrowser 控件，或者通过 OLE Automation 来利用 IE 的 COM 接口，也可以直接从 CHtmlView 派生类，以此得到所需信息。不过若只是检索少量内容，这未免有牛鼎烹鸡之嫌。那么有更简便的方法，就是使用 MSAA API 中的函数。

这只需使用一个 API，即：ObjectFromLresult 它的声明在 oleacc.h 中。一共 4 个参数：第一个是前一次调用成功后返回的 LRESULT 值，这个值在后面的示例中会有所说明；第二个是期望获取的 COM 接口标识符，这里是 IID_IHTMLDocument3；第三个是与之相匹配的属性值，一般置 0 即可；第 4 个就是文档对象指针，类型是与第二个参数相对应的 IHTMLDocument3 接口。关键代码如下：

```
POINT pos={-1,-1};
        GetCursorPos(&pos);
                UINT msg (RegisterWindowMessage ("
WM_HTML_GETOBJECT"));
        CWnd* pWnd(m_pDlg->WindowFromPoint(pos));
        LRESULT ret(-1);
        SendMessageTimeout (pWnd ->GetSafeHwnd (),
msg,0,0,SMTO_ABORTIFHUNG,
                MSG_POSTPONE,(PDWORD_PTR)&ret);
        CComPtr<IHTMLDocument3>pDoc(0);
    ret =ObjectFromLresult (ret,IID_IHTMLDocument3,0,
(LPVOID*)&pDoc);
        if(S_OK==ret)
        {
            CComPtr<IHTMLElement>pElem(0);
        ret=pDoc->get_documentElement(&pElem);
            CComBSTR text;
            ret=pElem->get_outerText(&text);
        }
```

代码的意图是获取鼠标点击下的当前页面，注意到函数 SendMessageTimeout 的最后一个返回参数，它就是上文所说的“前一次调用成功后返回的 LRESULT 值”。另外由于 WM_HTML_GETOBJECT 不是系统消息，因此需要注册。在得到 Document 对象接口，后面的工作就简单了，通过它获得 IHTMLElement 接口，再用 get_outerText 方法得到其中的内容即可。顺便提一句，用 get_documentElement 配合 get_outerText 方法得到的是网页完整的静态内容，与在 IE 浏览器的查看源文件菜单的结果一样。若想得到 Java 脚本添加的动态内容，还要换用 IHTMLDocument2 的 get_body 配合 get_innerText 方法得到解析后的动态结果，再替换先前静态内容中的 body 标签内的部分即可。另外两个方法：get_outerHtml 和 get_innerHtml 意义类似，只不过是附带着所有 HTML 标签而已。

为了验证程序的正确性，以此制作了一个简单的 E-mail 识别演示。正则表达式如下：(([a-z0-9] {1}) ([\\.a-z0-9_-] *) @ ([a-z0-9] +) (\\. ([a-z] +)) {1,3}) 可以识别 @ 符号后面两段、3 段或 4 段的电邮地址，编程时应附加 regex_constants::icase 属性以忽略大小写。使用时，在网页空白处按住 ALT 键+鼠标左键，保持 50 毫秒以上，即可识别此页所有 E-mail。这时有 1 秒左右的硬直状态，表示正在处理，不是死机，按下“保存”按钮，结果在 zzEmail.txt 中。所附代码在 VS2010+Windows 7 64 位系统下调试通过，不需安装其他 SDK，测试浏览器是 IE9。

（作者：申晓）

（上接第 92 页）

itor”加载该驱动程序后，只有第一次选择“Start Driver”时，才显示驱动程序中语句“KdPrint”宏中的内容（指 Checked 版，下同）；选择“Stop Driver”在在信息栏显示“ERROR (1052) : The driver is not in a state to accept this command.”。如果再次选择“Start Driver”，则在信息栏显示“ERROR (1058) : The driver is marked as disabled (Start=4) in its service database entry.”，并且不显示驱动程序中语句“KdPrint”宏中的内容。只有重新启动计算机，才能再一次显示驱动程序中语句“KdPrint”宏中的内容。

同样，在 WinDBG 中调试驱动程序时也是只能一次进入“DriverEntry”函数，显示驱动程序中语句“KdPrint”宏中的内容。只有重新启动虚拟机才能再次进入“DriverEntry”函数，显示驱动程序中语句“KdPrint”宏中的内容。

如果要正常显示驱动程序中语句“KdPrint”宏中的内容或多次进入“DriverEntry”函数，则需要在驱动程序中添加卸载函数，并在“DriverEntry”函数中添加“pDriverObject->DriverUnload = HelloDDKUnload;”语句。

7 结语

通过编写驱动程序实现了开机自动运行程序的目的，并且在 Windows 2003/XP 下测试通过。

参考文献

[1] Mark E.Russinovich, David A. Solomon. 潘爱民，译. 深入 Windwos 操作系统. 4 版. 北京：电子工业出版社，2007.

[2] 张帆，史彩成，等. Windows 驱动开发技术详解. 北京：电子工业出版社，2008.

（收稿日期：2013-02-25）

电脑系统维护经验与技巧

怎样处理大容量硬盘的分区问题

如今主流的硬盘容量越来越大，在 DOS 下使用一些传统的分区软件对硬盘进行分区时，有可能会无法识别正确的容量或是出现其他错误，这是硬件的兼容性问题。这里给用户提供两种办法，都可以轻松地给大硬盘分区，一是使用 Windows 系统安装光盘启动电脑，在系统安装程度过程中使用其自带的分区工具（就是蓝底白字的画面）进行分区，不仅速度快而且容易操作，兼容性也是最好的。二是将硬盘作为从盘挂到其他电脑上，使用“磁盘管理”工具即可分区。

怎样处理串口硬盘和光驱不能并存的问题

早期的带 SATA 接口的主板对 SATA 设备的支持不是很好，特别是对 SATA 接口的光驱，常常会发生找不到光驱、光驱无法引导问题。另外，因为系统安装等方面的问题，以前使用 SATA 硬盘、光驱时一般是将主板 SATA 接口设置为兼容模式，即将 SATA 接口映射到 IDE 通道上，将串口的 SATA 设备当做并口设备来使用。这种模式虽然兼容性较好，但有时也会发生串口和并口设备，或串口和串口设备占用共同的 IDE 通道，发生冲突的问题。解决方法有两个，一个方法是更换一下串口硬盘或光驱占用的主板接口，看在哪种接口组合下，硬盘、光驱能够和平共处。另一个方法是到 BIOS 中将 SATA 接口设置为 Enhanced Mode (增强模式)，但在此模式下可能需要使用整合了 SATA 驱动的光盘来安装操作系统。

怎样处理移动硬盘的分区无法读取的问题

如果在使用移动硬盘的过程中出现“缓存写入失败”错误，拔下数据线重新接上后，3 个分区只看到两个，而且系统会出现“假死”，拔下数据线就恢复正常。出现这样的问题，在“设备管理器”中对移动存储设备启用“写入缓存”策略后，可将需要移动的数据暂时放到缓和存中，等系统闲置时再执行实际的数据写入或读取操作。启用此策略可提高系统性能，但每次移除移动硬盘前必须在任务栏中手动卸载设备，以通知系统及时转移缓存中的数据。如果没有手动卸载设备或系统，正在后台转存数据时就强行拔掉移动硬盘，则很可能会出现“缓存写入失败”错误，还可能对硬件造成伤害。当然，移动硬盘数据线质量差、抗干扰能力弱、供电不足、硬盘或硬盘盒的硬件故障等原因也会造成“缓存写入失败”。建议用户更换质量更好的数据线或硬盘盒，在保证供电正常的情况下对移动硬盘进行读写，如果系统还是会卡或者要恢复数据，可尝试用 DiskGenius 修复一下移动硬盘的分区表。

如何屏蔽硬盘物理坏道

用 PartitionMagic 就可以处理。先用 PartitionMagic 中的“Check”命令来扫描磁盘，大概找出坏簇所在的硬盘分区，然后在 Operations 菜单下选择“Advanced/bad Sector Retest”.再通过 Hide Partition 菜单把坏簇所在的分区隐藏起来，这样就可以避免对这个区域进行读写。如果系统提示“TRACK 0 BAD, DISK UNUSABLE”，那么说明硬盘的 0 磁道出现坏道。这需要通过 Pctools9.0 等磁盘软件把 0 扇区 0 磁道屏蔽起来，最后用 1 扇区取代它就能修复。以 Pctools9.0 为例，运行 Pctools9.0 中的 de.exe 文件，接着选主菜单 Select 中的 Drive，进去后在 Drive type 项选 Physical，按空格选中它，再按 Tab 键切换到 Drives 项，选中 hard disk，然后回到主菜单，打开 Select 菜单，在出现的 Operation Table 中，选中硬盘分区表信息。找到 C 盘，该分区是从硬盘的 0 柱面开始的，那么，将 1 分区的 Beginning Cylinder 的 0 改成 1，保存后退出，重启后再重新分区、格式化即可。

怎样将 VCD 内容转存到硬盘

不需要安装什么软件就可以将 VCD 拷贝到硬盘中。用右键单击光驱，选择“打开”，可以看到一个名为“MPEGAV”的文件夹，这个文件夹的文件都是 DAT 的文件，复制到硬盘当中就行了（如果光盘本身没有问题或光驱读烂盘的能力超强的话就没有问题）。用 Windows 自带的播放器或者暴风影音之类的播放软件就可以播放。用户也可以将“.dbt”的后缀名改为“.mpeg”，这样就可以直接调用自带的播放器来播放了

怎样找出硬盘里重复文件

电脑使用一段时间后，硬盘中可能有很多重复文件，既不方便管理，又浪费硬盘空间，借助下面两个工具软件，可以快速查找并清理重复文件。

(1) FindDupFile

该工具软件特点如下：

1) 操作步骤简明

软件运行后，只需要“添加”需要查找的分区或目录，然

后点击“查找重复文件”软件就开始查找了。如果对结果有疑问，可以双击文件进行详细查看。

2） 清除方便

为了方便选择大量重复文件，软件提供了“勾选右边整列”功能。勾选文件后，点击“3”步中的删除可将勾选的文件删除。值得一提的是，考虑到删除重复文件后可能会产生很多空文件夹，软件还提供了删除它们的功能。点击左侧“4”右边的“查找”，即可快速查找源文件夹中的空文件夹，搜索完成后点击“查找”右边的“清除”，即可将空文件夹“消灭”干净。

(2) DoubleKiller

该软件具有以下两个鲜明的特点：

1） 限制选项多，过滤功能强

和 FindDupFile 相比，DoubleKiller 过滤选项设置丰富，过滤功能强。可根据需要设置扫描的类型文件（默认为所有文件）属性和大小等，以减小搜索范围，这样可以减少搜索时间。

2） 重复文件可删可移

相比 FindDupFile，DoubleKiller 在处理重复文件的方式上较为灵活，可选取前面的复件和后面的复件，选择后的项目会用“X”标示出来，再点击“删除选取的文件”即可将选择文件删除。为了安全起见，可以选择“移动选取的文件”，将重复文件移到另一位置。

如何禁止访问某个盘符

一个简单的方法是，在“开始”→“运行”中输入“gpedit.msc”，打开组策略，依次打开“本地计算机策略→用户配置→管理模板→Windows 组件→Windows 资源管理器”选项。在右边的设置窗口中，选择“防止从‘我的电脑’访问驱动器”，在这个选项单击鼠标右键，选择“属性”，接着弹出“防止从‘我的电脑’访问驱动器属性”设置窗口，在窗口中有 3 个选项，分别是“未配置”、“已启用”、“已禁止”。选择“已启用”，在下面就会出现选择驱动器的下拉列表，如果希望限制某个驱动器的使用，只要选中该驱动器就可以了。比如，用户要限制 C 盘的使用，选择“仅限制驱动器 C”。

如果用户希望关闭所有的驱动器，包括光驱等，可以选中“限制所有驱动器”。设置完毕后，当再打开驱动器时，就会弹出禁止打开驱动器的提示框。即便是使用 DIR 命令、运行对话框和镜像网络驱动器对话框，也无法查看驱动器的目录。

UIPower 助力新奥特完成软件界面的革新项目

新奥特视频技术有限公司是国内领先的数字媒体内容生产及运营的技术、服务提供商。其自主创新的各类产品及解决方案在各领域都有非常广泛的应用。技术产品包括：图文创作系统、非线性编辑系统、网络制播系统、虚拟演播系统等。应用行业遍及广播电视、党政机关、国防军队、交通能源、航空航天、教育培训、影视公司、企业集团等。典型的客户有中央电视台、北京电视台、上海电视台等。

七年前，各个软件企业都在追求软件功能的开发以解决从无到有的问题，大家都在比谁的软件功能更加强大，所以在添加功能时一般很少在意软件的用户体验。然而，最近的这几年软件用户不仅只是关注软件功能的多寡，而且更看重软件的易用性和可观度，因为毕竟再强大的功能都需要通过软件界面来进行呈现的。新奥特与客户接触最多的市场部和产品部纷纷接到客户对软件产品界面不满意的投诉和抱怨。董事长郑福双敏锐地感觉到软件产品的用户体验度的提升刻不容缓，软件界面一方面会直接影响到用户对软件的上手快慢与工作的效率，另一方面会影响到售前营销活动中的用户第一印象。于是成立软件界面革新项目小组，由公司技术总经理和市场总经理联合牵头，专门负责全公司上下的所有软件产品的界面革新工作。万事开头难，那么多系统，那么多代码，每次组织技术人员讨论界面改造方案时，老总们收到的都是些反对的意见。大家都担心代码量太大而存在很大的风险。然而，市场的需求摆在眼前，如何快速而顺利地完成此次重大的革新则显得至关重要。郑福双当机立断下了决策：借助外脑。

经过一段时间的多家界面供应商的选择，最终选定了上海勇进软件（UIPower.com）来承担此次所有生产线软件产品的界面革新工作。选择的理由是 UIPower 提供从 UI 的交互设计、视觉设计、界面开发到界面集成的整套流程的整体界面解决方案。目前国内做 UI 设计的公司居多，但他们都无法将 UI 设计真正做到软件程序中去，只能交给软件企业自行完成。而软件界面开发是一项专业要求高，工作量巨大的工作，要彻底做好界面必须借助界面工具，而国内能将工具和服务融为一体的，也只有 UIPower 了。

UIPower 经过三个月的努力，终于将新奥特的一款最核心最庞大的系统—喜马拉雅非线性编辑软件的界面改造完成了。

新奥特自发布带有新界面的喜马拉雅系统以来，获得了来自用户的一致好评，也获得了市场人员的在工作一线的客户良好的反馈。

UIPower 的 DirectUI 产品以界面与业务逻辑的彻底分离技术在各种需求的客户中真正实现了“让天下没有难做的界面”的企业使命。

书名：Objective-C 开发范例代码大全
ISBN：978-7-302-31364-9
定价：49.80 元
作者：（美）坎贝尔（Campbell, M.）著

本书采用“先提出问题，后提供解决方案”的方式讲解 Objective-C 编程中的核心内容，是 iOS 开发人员手中不可或缺的参考指南。

本书作者 Matthew Campbell 在培训 iOS 开发新手方面拥有非常丰富的经验，本书将向你展示如何使用 Objective-C 语言的独有特性以及 Foundation 框架的众多特性。此外，书中广泛提供了多种问题的解决方案，包括：使用 Xcode 进行应用开发、使用字符串、数字与对象集合、使用 NSArray、NSString、NSData 等 Foundation 类、使用线程、多核处理与异步处理、构建使用了日期、定时器与内存管理的应用、如何在其他平台上使用 Objective-C。

书名：Oracle Database 11g & MySQL 5.6 开发手册
ISBN：978-7-302-31031-0
定价：79.80 元
作者：（美）麦克劳克林(McLaughlin, M.) 著

本书规划了这两种平台之间无缝操作的程序设计策略和极佳实践方式。您可以学会如何迁移数据库、移植 SQL 代码、使用 Oracle 和 MySQL 数据库以及配置高效率的查询。在这本内容广泛的书籍中，还包括了安全、监控和调试方面的技巧。本书作者是 Oracle ACE，也是美国爱达荷州杨百翰大学计算机信息技术系教授，参与 Oracle 公司系列产品的研发已经有 20 年。

本书特色：理解 Oralce Database 11g 和 MySQL 5.6 的架构；在两个平台之间转换数据库，并确保事务完整性；创建表、序列、索引、视图和用户账户；编写并调试 PL/SQL、SQL*Plus、SQL/PSM 和 MySQL Monitor 的脚本；执行复杂查询并管理数字和日期的计算；合并来源表中的数据并设置虚拟目录。

书名：Java 7 编程高级进阶
ISBN：978-7-302-31362-5
定价：78.00 元
作者：（美）萨朗（Sarang,P.）著

本书包含了众多专家级编程技术，学习这些技术可以让你的 Java 水平上升至一个新的台阶。借助真实环境下的代码示例与详尽介绍，本书展示了如何充分利用 Java SE 7 的强大特性，讲述了如何设计多线程与网络应用程序，集成结构化的异常处理，使用 Java 类库以及开发基于 Swing 的 GUI 与 applet。另外，继承、泛型与各种实用类也在书中进行了介绍。

本书特色：创建自定义的类、方法、数组与操作符；使用条件语句控制程序流；处理多线程、网络及 I/O 编程；学习多线程中的新构造；使用枚举、注解与自动装箱；从错误、输入故障和异常中恢复；使用 Java Swing 构建轻量级的 GUI 与 applet；使用集合框架缩短开发时间；使用最新的 Java 类库与各种实用类。

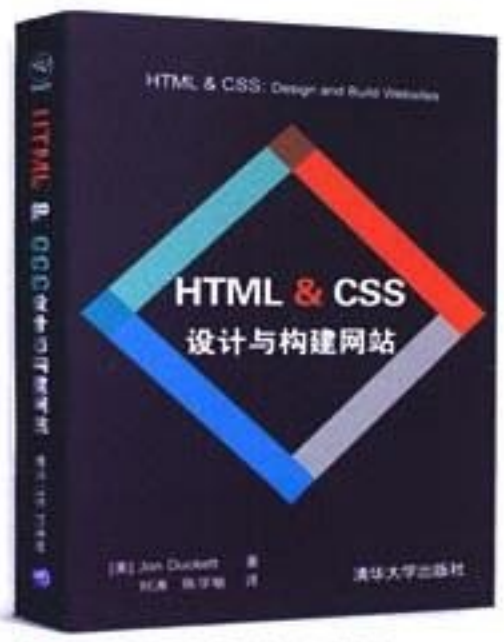
书名：App Store 创赢艺术——Apple 开发的赚钱机密
ISBN：978-7-302-30501-9
定价：59 元
作者：（美）坎塔尔季奇（Kantardzic, M.）著

iOS 平台的进入门槛不高，对于有志在移动应用领域有所表现的开发者来说，能很容易掌握开发技术，同时也能容易地将自己开发的应用投入到 iOS 平台上，包括移动游戏在内的移动应用创业将会是移动互联网领域非常有前景和极有商机的方向。

本书特色：App Store 目前和未来的趋势；竞争性研究的方法；应用开发的成本；App Store 的商业模式和客户期望；组织开发团队和外包；应用开发的各个阶段（包括软件发布与版本发布）；应用市场营销；在应用中集成社交网络（包括 Facebook、Twitter 和 Game Center 等）；在市场营销活动中极大限度利用社交媒体的有效手段；产品发布后的维护和支持选择。

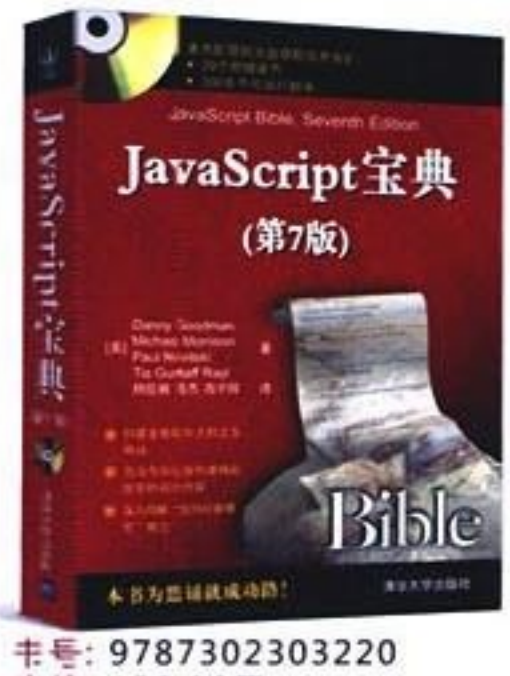
清华大学出版社
http://www.tup.com.cn

web开发攻略 私房大公开



书号: 9787302311034
定价: 59.80元

一直稳居Amazon Computer & Technology榜首！



书号: 9787302303220
定价: 128.00元

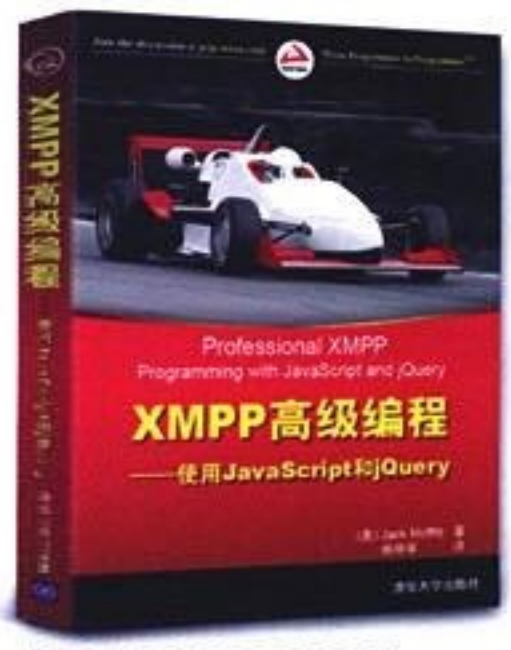
连续畅销十年

书号: 9787302245612
定价: 88.00元

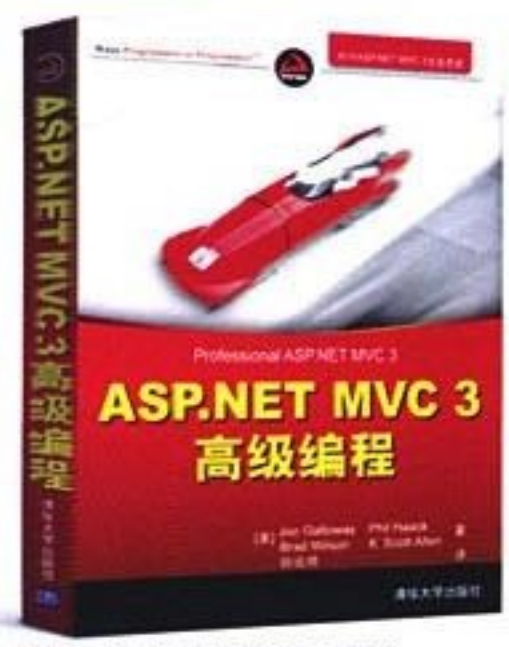
JavaScript入门必备，经典著作！

书号: 9787302283379
定价: 39.00元

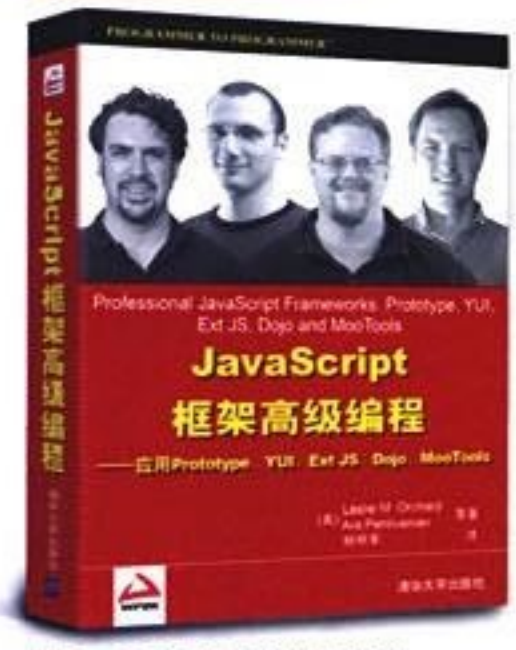
Web设计大师Dan Cederholm畅销力作



书号: 9787302256304
定价: 58.00元



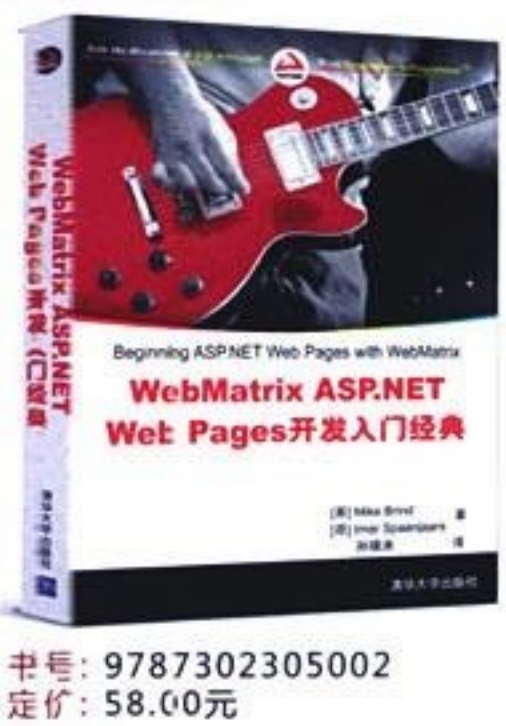
书号: 9787302286752
定价: 59.00元



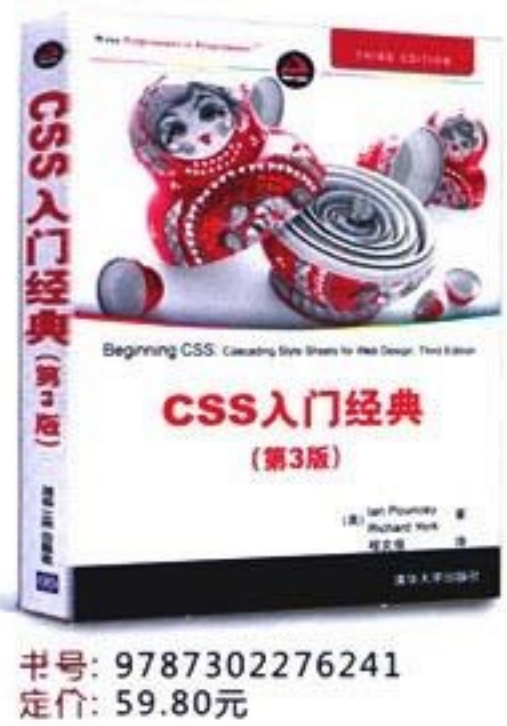
书号: 9787302247838
定价: 98.00元

书号: 9787302257073
定价: 79.80元

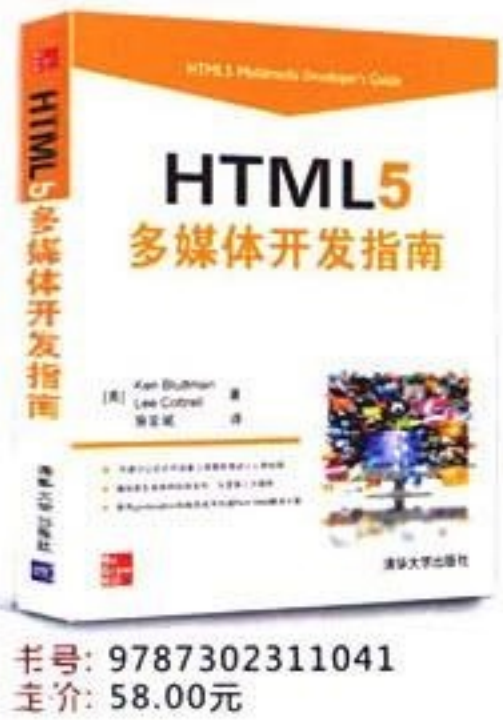
书号: 9787302251712
定价: 88.00元



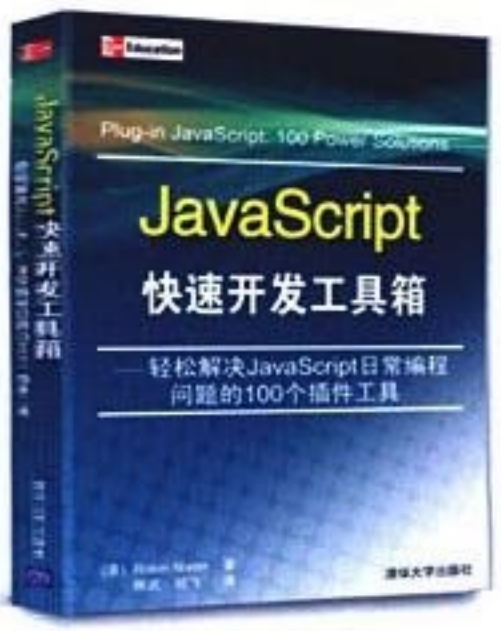
书号: 9787302305002
定价: 58.00元



书号: 9787302276241
定价: 59.80元



书号: 9787302311041
定价: 58.00元



书号: 9787302267034
定价: 59.00元

书号: 9787302274988
定价: 49.00元

书号: 9787302283324
定价: 88.00元

书号: 9787302255543
定价: 98.00元

GMGC

Global Mobile Game Confederation

第二届GMGC全球移动游戏大会

The 2nd Global Mobile Game Congress

“智能·创新·趣味”

Kevin Chou
Kabam CEO

岂永泉
岂乐网 CEO

胡泽民
91无线 CEO

刘勇
热酷创始人及CEO

王峰
蓝港在线 CEO

+100名演讲嘉宾同台论战

日本、美国、意大利、俄罗斯、新加坡、韩国、德国、中国等

+11国家或地区 850名移动游戏精英共司见证

+100家媒体，1000篇报道，总体传播PV突破20万

5折购票优惠码：
GMGC2013

时间：2013年3月6日
地点：北京·国家会议中心
主办：GMGC全球移动游戏联盟